PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

Universidade Federal do Rio de Janeiro

DISSERTAÇÃO DE MESTRADO

ELTON ALVES COSTA

CAMAW: um algoritmo de clusterização para múltiplas aplicações em redes de sensores sem fio

Rio de Janeiro
2016

Instituto de Matemática

Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
INSTITUTO DE MATEMÁTICA
INSTITUTO TÉRCIO PACITTI DE APLICAÇÕES E PESQUISAS COMPUTACIONAIS
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

Elton Alves Costa

CAMAW: um algoritmo de clusterização para múltiplas aplicações em redes de sensores sem fio

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Informática da Universidade Federal do Rio de Janeiro, como requisito parcial à obtenção do título de Mestre em Informática.

Orientadores: Prof[a]. Luci Pirmez, D. Sc.

Prof. Claudio Miceli de Farias D. Sc

Rio de Janeiro

2016

S237 Costa, Elton Alves C..
CAMAW: um algoritmo de clusterização para múltiplas aplicações em redes de sensores sem fio / Elton Alves Costa. -- 2016.
80 f.: il.

Dissertação (Mestrado em Informática) – Universidade Federal do Rio de Janeiro, Programa de Pós-graduação em Informática, Instituto de Matemática, Instituto Tércio Pacitti, Rio de Janeiro, 2016.

Orientadores: Luci Pirmez, Claudio Miceli de Farias.

1. Clusterização. 2. Ciência de aplicação. 3. Redes de sensores sem fio. 4. Múltiplas aplicações – Teses. I. Pirmez, Luci (Orient.). II. Universidade Federal do Rio de Janeiro. Programa de Pós-graduação em Informática. Instituto de Matemática. Instituto Tércio Pacitti. III. Título.

CDD:

Elton Alves Costa

CAMAW: um algoritmo de clusterização para múltiplas aplicações em redes de sensores sem fio

Dissertação de Mestrado submetida ao Corpo Docente do Programa de Pós-Graduação em Informática da Universidade Federal do Rio de Janeiro e à banca externa convidada como parte dos requisitos necessários para a obtenção do título de Mestre em Informática.

Aprovada em: Rio de Janeiro, _____ de _____ de _____.

_____

Prof[a]. Luci Pirmez – Orientadora
D. Sc., COPPE/UFRJ, Brasil.

_____

Prof. Claudio Miceli de Farias
D. Sc., UFRJ/PPGI, Brasil.

_____

Prof[a]. Priscila Machado Vieira Lima
Ph.D. Imperial College London, Grã-Bretanha.

_____

Prof. Luiz Fernando Rust da Costa Carmo
Dr., Université Toulouse III Paul Sabatier, França.

_____

Prof. Luiz Henrique Maciel Kosmalski Costa
Dr., Université Pierre et Marie Curie, França

Rio de Janeiro

2016

Dedico esse trabalho a toda a minha família, em especial à minha esposa Monica e meus filhos Rafael e Mariana.

AGRADECIMENTOS

Agradeço, acima de tudo, a Deus por ter me dado a vida, a saúde e também por ter sido meu conselheiro mais íntimo em toda a jornada até aqui.

À minha família, sobretudo à minha esposa Monica pela dedicação, pela paciência e pela disposição para cuidar de tudo, deixando-me livre, a maior parte do tempo, para me dedicar quase que exclusivamente ao mestrado. Aos meus filhos Rafael e Mariana por servirem de motivação para a conclusão desta importante etapa de minha vida, sempre me alegrando com seus sorrisos nos momentos difíceis. Aos meus sogros Severino e Natalina por nos acolherem em sua casa durante parte do mestrado para nos auxiliar nos cuidados com a caçula quando de seu nascimento.

Um agradecimento especial à minha orientadora, Profa. Dra. Luci Pirmez, primeiramente por permitir meu ingresso no programa, mas principalmente por sua dedicação, clareza e excelência com a qual conduziu minha orientação durante todo o período do mestrado. Percebo, melhor agora, a sabedoria que havia em cada palavra sua dirigida a mim. Agradeço-a também por sua compreensão em todos os momentos em que não correspondi ao esperado por conta de meus problemas pessoais.

Não há como agradecer ao meu orientador Prof. Dr. Claudio Miceli de Farias por tudo que fez por mim neste mestrado. Ele certamente foi além no seu papel de orientador e me ajudou literalmente “a carregar o piano”. Fico honrado por ter testemunhado o início de sua brilhante trajetória atualmente como professor adjunto do NCE. Percebo que o futuro NCE não poderia estar em melhores mãos.

Agradeço aos demais professores do PPGI que tiveram papel importante na realização deste trabalho, em especial a: Flavia C. Delicato por suas contribuições no artigo, críticas e revisões; ao Luiz F. Rust da Costa Carmo por seus ensinamentos; a Silvana Rosseto por suas críticas e sugestões; ao Adriano Cruz pelos importantes ensinamentos que contribuíram para realização deste trabalho.

Gostaria de ser capaz de agradecer aos amigos de labnet que facilitaram a minha jornada, mostrando atalhos e retirando obstáculos do caminho. Agradeço, especialmente, aos colegas Gabriel Caldas por, além de auxiliar na codificação do algoritmo, também ter disponibilizado seu trabalho para ser comparado ao meu; Igor dos Santos Leão por suas revisões; Profa. Emanuele N. L. de Figueiredo Jorge por compartilhar seu conhecimento e experiência e também todo pessoal do Labnet: Tiago França, Eduardo Hargreaves, Ítalo Cruz, Gabriel Martins, Alvaro Robles e Victo Pires; pois este trabalho também é fruto do esforço conjunto desta grande equipe.

Ao meu povo de Marinha, em especial a: SCNS Vicente R. M. Linhares por autorizar e solicitar o meu destacamento para realização deste curso de mestrado; Ao SCNS Dr. Alexandre S. Alves por seus conselhos e orientação; Ao SCNS Dr. Rodrigo da S. Moreira por auxiliar na revisão do trabalho; Ao meu chefe de grupo CF Adriano G. de Carvalho por sua orientação e também por possibilitar as condições dentro da marinha para a conclusão desta missão; Ao meu chefe de divisão CC Marcio de Melo Silva por ter feito tudo a seu alcance para ajudar na conclusão desta missão e também por suas importantes palavras de motivação e apoio; e por fim ao CT Edmar do Sacramento por me ajudar a lidar com toda a burocracia junto à Marinha.

“Não é a força, mas a perseverança que realiza grandes coisas.”
Samuel Johnson

RESUMO

COSTA, Elton Alves. **CAMAW**: um algoritmo de clusterização para múltiplas aplicações em redes de sensores sem fio. 2016. 80 f. Dissertação (Mestrado em Informática) – Programa de Pós-Graduação em Informática, Instituto de Matemática, Instituto Tércio Pacitti, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2016.

Nos últimos anos, os avanços nas áreas de sistemas micro-eletromecânicos aliados a um grande esforço em pesquisa têm auxiliado na disseminação das redes de sensores sem fio (RSSFs). Recentemente, o uso interdisciplinar das RSSFs tem adquirido destaque, o que vem a estimular o desenvolvimento de novos serviços e aplicações a serem suportados pelos nós sensores. Desde que as redes de sensores passaram a fazer parte do cotidiano das pessoas existem cada vez mais aplicações onde estes sistemas inteligentes são utilizados. Para possibilitar uma utilização eficiente da RSSF em um ambiente com múltiplas aplicações, este trabalho propõe um algoritmo de clusterização denominado "*Clustering Algorithm for Multiple Applications in a Wireless Sensor Network*" (CAMAW). O CAMAW permite à RSSF atender diversas aplicações simultaneamente a partir de agrupamentos de nós sensores criados com base na combinação entre as capacidades de sensoriamento dos nós e os requisitos de monitoramento das aplicações. Os principais benefícios do uso do CAMAW são: a eficiência energética da RSSFs, pois reduz o tráfego de mensagens de dados através da multiplexação de um mesmo tipo de monitoramento para diversas aplicações, partindo de um subconjunto de nós aptos a realizar o monitoramento requerido; e ser dinâmico, porque organiza os nós da RSSF em *clusters* que são capazes de lidar com a chegada e a saída de aplicações na rede. Os resultados dos experimentos demonstraram que em relação a outros trabalhos da literatura o CAMAW é mais energeticamente eficiente, pois possui desempenho superior quanto à duração da rede em todos os experimentos com múltiplas aplicações realizados.

Palavras-chave: Clusterização. Ciência de aplicação. Redes de sensores sem fio. Múltiplas aplicações. Compartilhamento de dados. Eficiência energética.

ABSTRACT

COSTA, Elton Alves. **CAMAW**: um algoritmo de clusterização para múltiplas aplicações em redes de sensores sem fio. 2016. 80 f. Dissertação (Mestrado em Informática) – Programa de Pós-Graduação em Informática, Instituto de Matemática, Instituto Tércio Pacitti, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2016.

In recent years, advances in the areas of micro-electromechanical systems combined with a great effort in research have led to a wide deployment of wireless sensor networks (WSNs). Recently, the interdisciplinary use of WSNs has gained prominence, stimulating the development of new services and applications to be supported by the sensor nodes. Since these sensor networks are starting to take part of our life, there are gradually more and more applications where these smart systems are used. To enable efficient use of WSN in an environment with multiple applications, this paper proposes a clustering algorithm called "*Clustering Algorithm for Multiple Applications in Wireless Sensor Network*" (CAMAW). CAMAW allows the network to attend several applications simultaneously from groups created based on the combination of the sensing capabilities of the nodes and the applications monitoring requirements. The main benefits of CAMAW are: the energy efficiency, since it reduces the data traffic by the multiplexing of same type of monitoring data between various applications, from a subset of nodes capable of performing the required monitoring; and be dynamic, because it organizes the WSN nodes in clusters that can deal with the arrival and departure of applications. The experiments results showed that CAMAW is more energy efficient than other algorithms because it outperforms them in terms of network life-time in all experiments performed with multiple applications.

Keywords: Clustering. Application-aware. Wireless sensor networks. Multiple applications. Energy efficiency.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface
ARM	Advanced Risc Machine
BS	Base Station
BCH	Backup Cluster-Head
CDMA	Code Division Multiple Access
CAMAW	Clustering Algorithm for Multiple Applications in WSAN
CH	Cluster-Head
CM	Cluster-Member
CTP	Collection Tree Procotol
DAG	Directed Acyclic Graphs
FND	First Node Dies
FTSP	Flooding Time Synchronization Protocol
HNA	Half Node Alive
IDE	Integrated Development Environment
LEACH	Low Energy Adaptive Clustering Hierarchy
LND	Last Node Dies
MAC	Medium Access Control
ME	Micro Edition
MEMS	Micro Electro Mechanical Systems
PCT	Pacotes de Controle Trafegados
PDT	Pacotes de Dados Trafegados
PF	Porcentagem de bytes ocupados em memória Flash
PR	Porcentagem de bytes ocupados em memória RAM
RSC	Rede de Sensores Compartilhadas
RSSF	Rede de Sensores Sem Fio
RSSI	Received Signal Strength Indicator
SCCH	Self-Configurable Clustering Mechanism
SDK	Software Development Kit
SHAAL	Smart Home and Ambient Assisted Living
SHM	Structural Health Monitoring
Sun SPOT	Sun Small Programmable Object Technology
TDMA	Time Division Multiple Access
TVR	Tempo de Vida da Rede

SUMÁRIO

1 Introdução

Os desenvolvimentos recentes das áreas de sistemas micro-eletromecânicos (do inglês, *Micro Electro Mechanical Systems* – MEMS), comunicação sem fio e *design* de baixo consumo energético têm permitido a operação de pequenos dispositivos sensores alimentados à bateria (GUPTA et al., 2013). As redes de sensores sem fio (RSSF) são compostas por centenas de milhares de nós sensores que conferem a esse tipo de rede capacidades distribuídas de processamento, sensoriamento e comunicação sem fios. Os nós sensores de uma RSSF normalmente são distribuídos por uma grande área geográfica de interesse (a área de monitoramento) que às vezes é de difícil acesso. Sendo assim, é desejável que tenham a capacidade de trabalhar de forma autônoma, pois a substituição das baterias dos sensores nestes casos se mostra uma tarefa indesejável, ou mesmo inviável. Os dados ambientais coletados por nós sensores a partir da área de monitoramento são então transmitidos até um ponto de saída da rede chamado nó sorvedouro, tipicamente utilizando um protocolo de comunicação de múltiplos saltos (*multi-hop*). O nó sorvedouro estabelece uma interface entre a RSSF e os usuários e aplicações (DELICATO 2005).

Exemplos de aplicações que podem se beneficiar do uso de RSSF são o monitoramento estrutural (do inglês, *Structural Health Monitoring* - SHM) (XU et al., 2004), o monitoramento do habitat (MAINWARING et al., 2002), da vida selvagem (LIU et al., 2004), ambiental (WERNER-ALLEN et al., 2006), monitoramento de condição de máquinas (GAO e FAN, 2006), sistemas de vigilância (ARORA et al., 2004), acompanhamento médico (SHNAYDER et al., 2005), rastreamento (CHEN et al., 2010), entre outros.

Apesar da crescente utilização das RSSF em diversas áreas, sensores são dispositivos passivos e altamente limitados em termos de energia, capacidade de processamento, armazenamento e comunicação, os quais basicamente coletam e disseminam dados, mas não atuam no ambiente. Já as Redes de Atuadores e Sensores sem Fio (RASSF) incluem também dispositivos atuadores, que são capazes de executar ações no ambiente físico em resposta a dados ou eventos físicos detectados pelos dispositivos sensores. Os nós atuadores são capazes de controlar iluminação, controlar fontes de calor ou aplicar forças de controle sobre algum material (VINYALS et al., 2010).

No nível de toda a RSSF, as políticas de eficiência energética exploram o conhecimento sobre o comportamento global da rede para realizar economia de energia. Tradicionalmente, essas políticas são implementadas em diferentes camadas da pilha de

protocolos da RSSF, como na camada de enlace de dados (XIONG et al., 2014), na camada de roteamento (GNAWALI et al., 2009) e na camada de aplicação (HEINZELMAN et al., 2000).

Segundo Bouhafs et al. (2006) a utilização de algoritmos de agregação e/ou fusão de informação é uma das possíveis estratégias para se realizar economia de energia. Algoritmos de fusão de informação exploram tanto a capacidade de processamento dos nós sensores quanto a redundância entre os dados coletados por eles, com o objetivo de reduzir as necessidades de transmissões, pois a transmissão de um bit pela rede pode consumir mais energia do que a execução de milhares de instruções (ANASTASI et al., 2009; LIU et al., 2010; POTTIE e KAISER, 2000). Dessa forma procura-se permutar o custo energético de comunicação pelo custo de processamento. Outra vantagem das técnicas de fusão de informação, além da economia de energia, é contribuir para aumentar a precisão e confiabilidade dos dados gerados pelos nós sensores.

Outra possível estratégia para a economia de energia é o uso das técnicas de clusterização (ABBASI e YOUNIS, 2007). A ideia geral da clusterização é a organização da RSSF em grupos chamados *clusters*. Cada grupo possui um líder chamado *cluster head* (CH) e um número de nós membros (do inglês, *cluster members* - CMs). Os nós membros reportam seus dados para o seus respectivos *Cluster Heads*. O papel do *Cluster Head* é agregar os dados dos sensores de seu *cluster* e enviá-los para uma estação central (nó sorvedouro) diretamente ou através de outros *Clusters Heads* da rede. (MITRA et al., 2012).

Pelo fato dos CHs usualmente transmitirem dados a maiores distâncias, eles consomem mais energia comparados aos nós membros de *cluster*. Portanto uma técnica comum em algoritmos de clusterização é reorganizar periodicamente a rede com objetivo de selecionar nós com maior energia remanescente, para assumirem o papel de CHs. Isso faz com que a carga de trabalho seja distribuída uniformemente entre os nós da rede. A clusterização permite reduzir a contenção de canal e as colisões de pacotes, a partir de esquemas de sincronização intra e inter *clusters* como, por exemplo, o TDMA e CDMA respectivamente (YOUNIS et al., 2006), resultando em uma maior vazão da rede por causa da maximização da utilização do canal sob alta carga (HUANG et al., 2013). A clusterização tem mostrado ser um método efetivo em prolongar o tempo de vida da rede por uma série de motivos. Entre eles é importante ressaltar que como a distância entre os membros dos *clusters* (vizinhos) e seu respectivo CH é, em geral, menor do que a distância entre os sensores e o nó sorvedouro, a rede economiza energia (ROCHA et al., 2012). Outro importante motivo, segundo Pantazis et al. (2013), é que também é possível economizar energia ao se realizar a fusão e agregação de

dados no nível do CH, reduzindo assim o total de pacotes a serem transmitidos em direção ao nó sorvedouro.

Uma desvantagem da maioria dos algoritmos de clusterização existentes tem a ver com o fato de que são desenhados para as RSSFs que, tradicionalmente, estão restritas a uma única aplicação alvo (KAPOOR et al., 2015; HSIEH et al., 2013) e organizam seus *clusters* levando em conta a distribuição geográfica dos nós definida por RSSI ou posicionamento GPS (HERMETO et al., 2013). Dessa forma, a execução destes algoritmos pode resultar, segundo Bruckner et al. (2008), em *clusters* que embora compostos por nós geograficamente próximos uns dos outros, realizam monitoramento de ambientes completamente distintos, resultando em dados com baixa correlação como, por exemplo, o monitoramento em lados opostos de um muro. Isso pode distorcer a visão do usuário sobre o que acontece de fato no ambiente e levar a imprecisões na compreensão dos fenômenos da região monitorada. Além disso, ao não se considerar os requisitos das aplicações no procedimento de organização da rede, perde-se a oportunidade de realizar um pleno compartilhamento dos recursos da RSSF, o que pode representar um desperdício de energia.

Nos últimos anos, as RSSFs têm passado por diversas mudanças que impactaram o projeto e operação dessas redes. Dentre elas destaca-se o surgimento das Redes de Sensores Compartilhadas – RSC (do inglês, *Shared Sensor Networks* - SSN) (EFSTRATIOU et al., 2010) as quais, ao invés de assumir um projeto tradicional de RSSF desenhada para suportar uma única aplicação alvo, permitem que a infraestrutura de sensoriamento e comunicação seja compartilhada por múltiplas aplicações que podem pertencer a diferentes usuários, otimizando assim a utilização de recursos. Portanto, as RSCs podem ser entendidas como infraestruturas integradas de sistemas físicos e eletrônicos que podem servir a múltiplas aplicações (EFSTRATIOU et al., 2010).

Apesar de uma série de potenciais vantagens de um projeto de rede voltado para RSC, entre elas o rápido desenvolvimento e implementação de aplicações que podem ser atreladas à infraestrutura de sensoriamento (LEONTIADIS et al., 2012), a adoção de RSC apresenta novos desafios como a alocação de recursos, o gerenciamento da rede e a segurança (EFSTRATIOU et al., 2010). Um dos desafios que surgem diz respeito ao consumo de energia, que pode aumentar em consequência tanto do acréscimo de processamento, quanto do envio de mensagens relacionados ao maior número de aplicações (FARIAS et. Al. 2012). Para Müller e Alonso (2006) o suporte a múltiplos usuários é um dilema entre uma operação eficiente da rede (com redução de tráfego e de processamento, eliminando redundâncias em

termos de medições e mensagens) e o crescente número de requisições de usuários independentes que devem ser atendidos (cada um potencialmente interessado em um conjunto diferente de sensores e taxa de aquisição). Os trabalhos de Leontiadis et al. (2012), Haghighi e Cliff (2013), Khalid et al. (2014) e Caldas et al. (2015) demonstram que a clusterização têm sido um dos métodos mais utilizados para lidar com este dilema.

Apesar de existir a possibilidade de captação de energia por meio de fontes naturais, como sugerem Roundy e Frechette (2005), essas estratégias apresentam problemas relacionados à confiabilidade e estabilidade das fontes ao longo do tempo. Desta forma, é essencial que essas redes estejam conscientes de sua própria quantidade de energia disponível e sejam capazes de lidar de forma inteligente com o seu consumo para maximizar o seu tempo de vida.

Vários desafios surgem ao se projetar algoritmos de clusterização voltados para múltiplas aplicações. Segundo Kappor et al. (2015) quando mais de uma aplicação utiliza uma mesma RSSF o gerenciamento dos recursos dos nós se torna o grande desafio: Diferentes aplicações podem ter diferentes áreas-alvo, diferentes interesses de monitoramento (acerca do tipo dados coletados), e diferentes taxas de coleta e transmissão de dados. Na abordagem de múltiplas aplicações a rede é organizada em agrupamentos de nós sensores que realizam tarefas de monitoramento para mais de uma aplicação. Sendo assim, nós sensores podem ser remanejados de um *cluster* para outro ao longo do tempo, de acordo com a necessidade das aplicações. Segundo os trabalhos de Takavoli e Kansal (2010), Farias et al. (2013) e Gao et al. (2015), o sensoriamento deve ocorrer considerando-se o interesse comum entre as aplicações, para que se possa evitar a repetição desnecessária de tarefas. Além disso, cada aplicação em geral está associada a um modelo específico de entrega de dados (HAGHIGHI e CLIFF, 2013). O que torna importante que, em um ambiente que contempla múltiplas aplicações simultâneas, a RSSF possua as características de ser configurável, dinâmica e flexível para prover, a qualquer tempo, a necessidade de monitoramento requerida por cada aplicação.

1.1 Objetivos

Este trabalho propõe um algoritmo de clusterização ciente das aplicações, denominado “Clustering Algorithm for Multiple Applications in a Wireless sensor network” (CAMAW). O CAMAW agrupa os nós com base na área de interesse e nos requisitos das aplicações, organizando a RSSF de forma hierárquica, possibilitando a sua utilização por aplicações concorrentes, com o objetivo de promover o compartilhamento de recursos da rede. O algoritmo faz com que os nós sensores sejam capazes de atender diversas aplicações

simultaneamente a partir de agrupamentos criados com base na combinação entre as capacidades de sensoriamento dos nós e os requisitos de monitoramento das aplicações. Em outras palavras, o CAMAW é um algoritmo de clusterização que promove um uso racional dos recursos da rede ao: (i) agrupar os nós sensores estritamente segundo os requisitos das aplicações, restringindo tanto a área do campo de monitoramento à área de interesse das aplicações, quanto o conjunto de nós da rede a um subconjunto de nós aptos a atender às aplicações; e (ii) permitir o compartilhamento dos recursos da rede a partir de sua capacidade de identificar redundâncias entre os requisitos de monitoramento das aplicações e utilizá-las como uma oportunidade para reduzir o esforço de monitoramento e de comunicação. A partir destas características, o CAMAW procura atender às necessidades de monitoramento de múltiplas aplicações de uma forma energeticamente eficiente.

1.2 Organização do Trabalho

Este trabalho está organizado em 6 capítulos. O capítulo 2 apresenta os conceitos básicos que são importantes para a compreensão desta proposta. O capítulo 3 descreve sucintamente alguns trabalhos relacionados, enfatizando as similaridades, as vantagens e desvantagens em relação a este trabalho. O capítulo 4 descreve a proposta do algoritmo e detalha os seus procedimentos. O capítulo 5 detalha o ambiente de implementação desta proposta. O capítulo 6 detalha os experimentos realizados para avaliar o algoritmo. O capítulo 7 conclui o trabalho e aborda questões para trabalhos futuros.

2 Conceitos Básicos

Este capítulo descreve os conceitos básicos relevantes para a compreensão deste trabalho. Este capítulo está organizado como descrito a seguir: A seção 2.1 apresenta os conceitos básicos sobre as redes de sensores sem fio; A seção 2.2 apresenta os conceitos básicos sobre as redes de sensores compartilhadas; A seção 2.3 descreve as características dos Algoritmos de Clusterização; A seção 2.4 traz uma pequena conclusão acerca das contribuições de cada conceito básico para a proposta.

2.1 Redes de Sensores Sem Fio

Uma rede de sensores sem fio (RSSF) é composta por nós distribuídos em uma região de interesse, com o objetivo de coletar informações ambientais. Estes nós possuem capacidade de sensoriamento, processamento, armazenamento e comunicação. As tarefas de extração e transmissão dos dados da rede são realizadas de forma colaborativa entre os nós sensores. Esta transmissão é realizada em direção a um ou mais pontos de saída da rede, chamados de nós sorvedouros, para serem analisados e adicionalmente processados. Tais sensores podem ser instalados com um posicionamento pré-definido ou de maneira ad hoc dentro da área alvo (DELICATO, 2005).

O custo e a capacidade inerentes a cada nó são limitadores para o tamanho e abrangência das RSSFs. Entretanto, os recentes avanços na tecnologia indicam que o poder de processamento e a memória são restrições temporárias nas RSSFs e tendem a desaparecer com o desenvolvimento das técnicas de fabricação (HE et al., 2004). A restrição de energia, por outro lado, permanece como uma questão crítica pelo fato de que a tecnologia de baterias não evolui no mesmo ritmo que o hardware utilizado nos dispositivos sensores e também por causa da dificuldade encontrada na manutenção da RSSF para reabastecimento de energia por recarga ou por substituição das baterias dos nós. Geralmente a intervenção humana na manutenção das RSSFs é arriscada, ineficiente e até mesmo impossível devido a enorme quantidade de dispositivos sensores e também pelo fato da região de interesse muitas vezes ser remota, inóspita ou de difícil acesso, tais como florestas, oceanos, desertos, vulcões, entre outras (ROCHA 2012). Portanto os sensores devem funcionar de forma autônoma e serem capazes de funcionar sem a intervenção humana por longos períodos de tempo. Mais importante ainda é que todas as etapas do projeto da RSSF e seu funcionamento devem considerar a questão energética.

Em relação à topologia, a rede deve possuir uma estrutura dinâmica e auto-organizável para comportar as situações onde, por exemplo: sensores podem ter seus recursos energéticos

esgotados e em consequência disso terem de deixar a rede; novos sensores podem ser incorporados durante o ciclo de vida da rede; sensores podem ser temporariamente desligados para economia de energia; sensores devem perceber alterações do ambiente no qual se encontram instalados; sensores devem atender a mudanças nos interesses de monitoramento das aplicações.

Com relação ao *hardware* dos nós sensores, as RSSFs podem ser classificadas em homogêneas, onde todos os nós são iguais; ou heterogêneas, caso os nós sejam diferentes em relação aos atributos físicos, tais como os dispositivos sensores que o nó carrega, seu poder de processamento e também seu consumo e reserva de energia.

Outro aspecto importante do funcionamento das RSSFs é a questão do roteamento. Existem vários protocolos diferentes, dos quais deve-se escolher o mais adequado à aplicação da rede. Normalmente estes protocolos têm a característica de ser *single-hop* ou na maioria dos casos, *multi-hop* e podem possuir mecanismos de agregação dos dados dentro da rede. Estes protocolos também ativam ou desativam os nós de maneira seletiva (DELICATO 2005). O resultado final deve ser a informação fluindo da origem dos dados até o seu destino de forma eficiente.

Com relação à sincronização os nós da RSSF devem estar sincronizados para que a partir dos dados reportados por eles seja possível realizar inferências coerentes a respeito do ambiente sendo monitorado (SUNDARAMA et al., 2005); já que nas RSSFs os dados provenientes dos diversos sensores geralmente são aglomerados com o objetivo de formar um único resultado significativo. Também é desejável que o protocolo de sincronização adotado

seja capaz de lidar com a mobilidade e a escalabilidade da RSSF, considerando-se que os nós sensores possuem recursos limitados e ficam localizados em ambientes não confiáveis e com alta perda de pacotes. Segundo Sundarama (2005) estes protocolos podem ser classificados quanto a sua precisão, acurácia, custo e complexidade. Em RSSFs com múltiplas aplicações tais características devem ser levadas em conta no procedimento de escolha do protocolo de sincronização mais adequado ao conjunto de aplicações em execução na rede.

Com relação às aplicações, segundo Borges et al.(2014), elas são a origem de uma série de requisitos impostos às RSSFs. Um deles é o modelo de entrega de dados, que varia conforme a natureza da aplicação como, por exemplo, se de tempo-real ou não, se tolerante a atraso ou não, se de missão-crítica ou não, entre outros. O autor descreve quatro modelos básicos de entrega de dados para as RSSFs: *Event-driven, Query-* ou *Demand-driven,*

Continuous based, *Time-driven.* O modelo *Event-driven* está geralmente associado a aplicações intolerantes a atraso (tempo-real) e de missão-crítica. Como consequência a detecção de eventos torna-se uma atividade bastante importante para o sucesso da aplicação. Neste tipo de modelo de entrega, os dados que fluem dos nós sensores são altamente correlacionados, o que leva a uma grande quantidade de amostras redundantes. Já no caso do modelo *Query* ou *Demand-driven* a maioria das aplicações são interativas, tolerantes a atraso e de missão-crítica. Neste modelo as consultas são realizadas sob demanda com o objetivo de economizar energia. É similar ao modelo *Event-driven*, mas diferem pelo fato de que no *Event-driven* os dados são "empurrados" pelos nós da rede em direção ao nó sorvedouro, enquanto que no *Query-driven* os dados são "puxados" pelo nó sorvedouro. No modelo *Continuous based* os dados são enviados em direção ao nó sorvedouro de maneira contínua e a uma taxa pré-definida. Neste modelo, diferentes tipos de tráfego de dados podem coexistir, mas com requisitos específicos diferenciando cada fluxo. Por fim, no modelo *Time-driven*, os nós sensores coletam e reportam seus dados periodicamente, sendo que o período entre leituras consecutivas é conhecido como **taxa de amostragem**. Neste modelo normalmente há pouco processamento sobre os dados ao nível dos nós e mais ao nível do nó sorvedouro.

2.2 Redes de Sensores Compartilhadas

Uma rede de sensores compartilhada (RSC) é uma RSSF que provê uma infraestrutura flexível e capaz de suportar o compartilhamento de recursos entre diversas aplicações. Os usuários submetem novas aplicações para a rede através do nó sorvedouro. Estas são instaladas dinamicamente em diferentes momentos, de acordo com a demanda dos usuários. Além disso, diferentes aplicações podem possuir diferentes prioridades, de acordo com a importância de uma em relação às outras. Uma rede de sensores compartilhada trabalha com uma infraestrutura altamente flexível, que suporta diferentes níveis de compartilhamento de recursos entre as aplicações. Por exemplo, múltiplas aplicações podem compartilhar: uma unidade de sensoriamento em um único nó sensor; um nó sensor contendo múltiplas unidades de sensoriamento e também uma rede de nós sensores executando diferentes aplicações. Aplicativos podem ser implantados de forma dinâmica em momentos diferentes, com base na demanda do usuário. Um nó pode ser equipado com uma ou mais unidades de sensoriamento, microprocessador, rádio e tudo isso ser compartilhado entre as aplicações da rede. A RSC representa a total dissociação de aplicações e infraestrutura física de sensoriamento e transmissão (EFSTRATIOU et al., 2010).

2.3 Algoritmos de Clusterização

A clusterizacão é uma das técnicas mais populares para tornar as RSSFs energeticamente eficientes (AFSAR E TAYARANI 2014; KATIYAR, CHAND, SONI, 2011; ABBASI E YOUNIS, 2007; YOUNIS, YOUSSEF, ARISHA, 2003). Além disso, o agrupamento dos nós em *clusters* pode prover outros benefícios de acordo com Afsar e Tayarani (2014) como, por exemplo, (i) escalabilidade; (ii) redução do tamanho e da complexidade da tabela de roteamento; (iii) economia de largura de banda; (iv) estabilidade da topologia da rede; (v) coordenação pelo CH das atividades dos sensores no *cluster* de forma que possa ocorrer: a ativação e desligamento dos nós para a economia energia, a prevenção de colisões no acesso ao meio e a limitação da redundância dos dados coletados; (vi) aplicação de métodos de fusão no CH para a redução do número de pacotes transmitidos; (vii) tolerância a falha; (viii) balanceamento de carga; (ix) aumento da conectividade pois basta apenas que exista ao menos um caminho entre cada CH e o nó sorvedouro para se alcançar a conectividade da rede, ao invés de entre cada nó da rede e o sorvedouro; (x) redução do atraso de roteamento; (xi) utilização de modos de economia e baixo consumo.

A ideia geral de esquemas de clusterização é a organização da rede em uma estrutura hierárquica em dois ou mais níveis, formada por grupos (*clusters*) onde cada grupo possui um líder CH. Este, por sua vez, também pode possuir um líder CH e assim por diante em uma arquitetura de múltiplos níveis geralmente utilizada para rotear os dados coletados até o(s) no(s) sorvedouro(s) em uma comunicação de múltiplos saltos (*multi-hop*), ou diretamente (*single-hop*), dependendo de quantos níveis de CH (AFSAR et al., 2014). O nó sorvedouro é um dispositivo computacionalmente poderoso, que não possui restrições de recursos, e que atua como ponto de entrada para os requisitos das aplicações e pontos de encontro para os dados coletados pelos sensores. Além de encaminhar os dados para o nó sorvedouro, o CH é responsável também por coordenar a comunicação *intra-cluster* e realizar a fusão dos dados recebidos pelos sensores.

A eleição do(s) papel de CH(s) entre os nós, portanto, é uma questão importante em métodos de clusterização, já que as funcionalidades delegadas ao CH geram um consumo maior de energia do que nos demais nós pertencentes ao *cluster*. Um CH pode ser previamente atribuído pelo desenvolvedor da RSSF, eleito pelo nó sorvedouro ou ainda eleito dinamicamente pelos sensores. Os algoritmos de clusterização também podem ser classificados em termos do processo de escolha do CH em centralizado, se definido pelo nó sorvedouro, ou distribuído se eleito pelos nós. Mas segundo Afsar et al. (2014) como o

esquema centralizado não é escalável e portanto não adequado às RSSFs, a maioria dos algoritmos de clusterização existentes é do tipo distribuído. Geralmente em RSSFs homogêneas, cujos sensores possuem igual capacidade em termos de computação, comunicação e energia, os CHs são selecionados entre os nós sensores da rede e frequentemente são excluídos das atividades de sensoriamento para reduzir o seu consumo de energia (ABBASI e YOUNIS, 2007). Em redes homogêneas geralmente realiza-se o rodízio do papel de CH entre os nós para que haja balanceamento do consumo de energia pela rede. Entretanto, nas RSSFs heterogêneas os CHs são geralmente selecionados entre os nós que possuírem as maiores capacidades em termos de energia. Este fato impõe restrições à estratégia de clusterização, pois é preciso relativizar tais capacidades no processo de escolha de CHs, pois estes nós diminuem a necessidade das constantes rodadas para troca de papéis no decorrer do tempo.

Outra questão importante em algoritmos de clusterização é a tolerância a falhas em nós CH (ABBASI; YOUNIS, 2007), usualmente necessária nas aplicações para evitar a perda de dados importantes. Como as RSSFs são geralmente empregadas em ambientes hostis, os CHs podem falhar, ter mau funcionamento ou até mesmo dano físico. O modo mais intuitivo de recuperar uma falha de um CH é reclusterizando a rede. Entretanto, a reorganização não gera somente consumo de recursos, mas interfere também na operação normal em curso. Técnicas contemporâneas de tolerância a falhas incluem atribuição de CHs substitutos em caso de falha de um CH, cuja seleção dos nós que serão substitutos dos CHs geralmente é realizada durante o funcionamento normal da rede.

Recentemente, por conta da constante busca por eficiência e melhor aproveitamento da infraestrutura das RSSFs, os algoritmos de clusterização tem sido utilizados como uma das técnicas, conforme descrito em Khan et al. (2015), para compartilhar a infraestrutura da rede entre várias aplicações diferentes. A estrutura da rede formada por agrupamentos de nós CMs em torno de nós líderes CHs favorece a inclusão, nestes líderes, de tarefas relativas à organização e monitoramento voltado para múltiplas aplicações. Dessa forma, é importante que os novos algoritmos sejam flexíveis para suportar os requisitos de diferentes aplicações. Devem operar, assim como no trabalho de Haghighi e Cliff (2013), com diferentes paradigmas operacionais ou modelos de entrega de dados (BORGES et al., 2014), com o objetivo de tratar uma ampla variedade de aplicações com necessidades distintas de monitoramento.

2.4 Conclusão

Neste capítulo foram apresentados os conceitos básicos que são importantes para o entendimento da proposta deste trabalho. Foram descritas as principais características e funcionalidades das RSSF, dos algoritmos de clusterização para RSSFs e também das Redes de Sensores Compartilhadas (RSCs), que são um tipo especial de RSSF. Foi descrito também como estes métodos auxiliam na economia de recursos da rede. Todos esses conceitos importantes colaboraram, em alguma medida, para a elaboração e implementação da proposta contida neste trabalho.

3 Trabalhos relacionados

Com o crescente interesse em promover um melhor aproveitamento da infraestrutura de monitoramento das RSSFs em ambientes multipropósito que contemplam diversas aplicações, têm surgido recentemente trabalhos voltados para as RSSFs com foco no tratamento de múltiplas aplicações. Nestes trabalhos algumas técnicas também utilizadas no CAMAW tem se destacado na promoção da eficiência da rede neste tipo de ambiente, como a clusterização e o compartilhamento dos dados entre as aplicações. Este capítulo se dedica a descrever alguns destes trabalhos. Para cada trabalho relacionado, são descritas as características gerais, semelhanças e diferenças de cada abordagem em relação à presente proposta, ressaltando-se, comparativamente, as vantagens do CAMAW. Exemplos de trabalhos que se enquadram nesses critérios são Leontiadis et al. (2012), Haghighi e Cliff (2013), Khalid et al. (2014), Caldas et al. (2015), Izadi et al. (2015), Takavoli et al. (2015), Gao et al. (2015) e Farias et al. (2013).

Em Leontiadis et al. (2012) é proposto o SenShare, um *middleware* voltado para as RSCs que tenta resolver os desafios técnicos que surgem a partir do nível da rede por meio da construção de redes de sensores *overlay*. Tais redes o*verlay* não são apenas responsáveis por fornecer os membros mais apropriados para executar as tarefas das aplicações, mas também por isolar o tráfego de rede de uma aplicação alvo do tráfego de rede gerado por outras aplicações ou pelos mecanismos que suportam a rede *overlay*. Para alcançar o objetivo de isolamento de tráfego, o SenShare estende cada pacote de dados da aplicação em tempo de execução com um cabeçalho de roteamento da aplicação que possui 6 bytes de comprimento, embora todas as mensagens da rede sejam ainda mantidas no formato do padrão IEEE 802.15.4. Uma vez que os nós sensores de um *cluster* podem estar localizados em qualquer lugar dentro da rede, os nós com tarefas atribuídas e vizinhos físicos que podem se comunicar com mensagens de um salto, são colocados então em um mesmo *cluster*. Isto resulta geralmente em um dado número de *clusters* que estão isolados uns dos outros. Para a construção de uma RSSF a partir destes *clusters*, como sendo uma única rede conectada e específica da aplicação, conexões virtuais entre os *clusters* precisam ser estabelecidas com a ajuda dos nós que não estão executando as tarefas da aplicação alvo. Conexões virtuais entre *clusters* são geradas de forma incremental em três etapas consecutivas, que são: 1) identificar os nós que estão nas bordas de um *cluster* de nós conectados, 2) descobrir caminhos ótimos a partir dos nós selecionados na etapa anterior que ligam o *cluster* local a outros *clusters* e 3) assegurar que todos os *clusters* estejam conectados entre si e sejam capazes de acessar o nó

sorvedouro da rede. No SenShare diversas instâncias da mesma RSSF podem executar isoladas, uma por aplicação. Por outro lado no CAMAW todas as aplicações executam em uma única instância, o que lhe permite encontrar e eliminar redundâncias de sensoriamento e comunicação, considerando-se a comunalidade de requisitos entre as aplicações.

Outro trabalho, Khalid et al. (2014), propõe o *smart home and ambient assisted living* - SHAAL, uma estrutura de *middleware* para a virtualização de rede em ambientes de casas inteligentes. O SHAAL é baseado na virtualização de rede de sensores, o que permite que várias aplicações pertencentes ao domínio de cuidados de saúde sejam executadas em uma rede com nós heterogêneos. No SHAAL, uma única aplicação pode ser distribuída ao longo de um dado número de *clusters*, e um mesmo nó é capaz de participar de vários *clusters*. Além disso, o compartilhamento da infraestrutura é possível graças a uma camada de abstração que reside em cada nó sensor. O gerente virtual, ou seja, o núcleo do *middleware* reside no nó sorvedouro ao qual todos os nós da rede estão conectados. O gerente virtual é o responsável pelo gerenciamento do *middleware* e decide, por exemplo, os agrupamentos de nós que atenderão a cada aplicação. Estes são formados com base nos requisitos das aplicações expressos em termos do tipo de dado a ser coletado pelos sensores como, por exemplo, batidas do coração, pressão sanguínea, detecção de queda, entre outros. Entretanto, diferentemente do CAMAW, somente a infraestrutura de nós sensores é compartilhada, enquanto as atividades de monitoramento da aplicação permanecem isoladas. O SHAAL de forma semelhante ao CAMAW também organiza a rede de acordo com a entrada e saída de aplicações. Entretanto, o CAMAW irá, sempre que possível, promover o compartilhamento de dados de monitoramento entre as aplicações. Ou seja, na medida em que as aplicações podem possuir interesses comuns de monitoramento, o CAMAW identifica tais situações como oportunidades para se reduzir o número de monitoramentos, compartilhando seus resultados entre as diversas aplicações interessadas. No SHAAL, ao contrário do CAMAW, pode haver a ocorrência de monitoramentos redundantes, que são repetidos desnecessariamente em número de vezes igual ao número de aplicações com interesses semelhantes, o que é claramente menos eficiente em relação ao consumo de energia.

Em Haghighi e Cliff (2013) é proposto o Sensomax, um *middleware* que oferece uma solução dinâmica de software, capaz de realizar a distribuição de tarefas pela RSSF e de coletar dados de uma forma integrada. Assim como o CAMAW, o Sensomax provê uma solução fim-a-fim onde as aplicações concorrentes programam e atualizam a RSSF conforme seus requisitos, permitindo a alteração e reprogramação das tarefas em tempo de execução. O

Sensomax trabalha dividindo a rede em múltiplos agrupamentos de nós, vinculados a uma ou mais aplicações. As aplicações podem potencialmente utilizar *clusters* sobrepostos, na medida em que cada *cluster* lida com os requisitos específicos de cada aplicação. Os nós, que são os atores da rede, atuam de acordo com os papeis de CH ou CM, assumido no contexto de cada aplicação, sendo que o CH de uma determinada aplicação pode atuar como CM em outra aplicação. Além disso, o Sensomax descentraliza o controle das aplicações para os CHs em detrimento do nó sorvedouro. O Sensomax, assim como o CAMAW, foi desenhado para lidar com vários paradigmas operacionais: *Data*, *Event*, *Timing* e *Query-driven*. A desvantagem do Sensomax frente ao CAMAW é que apesar do Sensomax promover o compartilhamento de recursos, visto que um mesmo nó pode trabalhar para mais de uma aplicação (como CM ou CH), os agrupamentos da rede pertencentes a diferentes aplicações não compartilham dados entre si como uma forma de evitar a repetição desenecessária de tarefas. Além disso, clusterizar a rede empilhando uma nova camada de agrupamentos a cada nova aplicação exige mais recursos em termos de memória e processamento dos nós, além de um tráfego maior de mensagens (de dados e de controle). O CAMAW por outro lado, estabelece uma organização da rede formada por uma camada única de agrupamentos que procuram conciliar os interesses comuns de monitoramento de diversas aplicações. Tal organização é capaz de adaptar-se conforme as aplicações entram e saem da rede. As características do CAMAW permitem que ele seja capaz de minimizar o esforço de organização e operação da rede reduzindo as tarefas de monitoramento e de comunicação.

O trabalho de Caldas et al. (2015) propõe o S-LEACH, um algoritmo de clusterização para redes de sensores compartilhadas, pois é projetado para lidar com várias aplicações compartilhando simultaneamente a mesma infraestrutura de RSSF. Sendo assim, no S-LEACH a formação dos *clusters* é realizada com o objetivo de encaminhar os dados para múltiplas aplicações, de forma a transmitir esses dados uma única vez. Além disso, ao considerar um cenário de aplicações compartilhadas executando sobre uma infraestrutura de sensores em comum, os nós CH do S-LEACH se valem do uso de algoritmos de fusão de dados projetados especificamente para uso em ambiente de redes de sensores compartilhadas, em vez de utilizar algoritmos de fusão de dados tradicionais. No S-LEACH o procedimento de agrupamento dos nós não considera as aplicações em execução, o que significa que ele primeiro organiza toda a rede e, apenas após, recebe as requisições das aplicações. Ou seja, somente após a rede completamente agrupada, o algoritmo promoverá o compartilhamento dos dados de monitoramento entre as aplicações. Enquanto que o CAMAW somente organiza

os agrupamentos conforme os requisitos das aplicações, restringindo a área de monitoramento à área de interesse das aplicações e minimizando os esforços de agrupar, monitorar e transmitir. O CAMAW também evita a participação nos agrupamentos de nós que são desnecessários ou inúteis ao monitoramento pretendido.

Em Izadi et al. (2015) os autores apresentam um algoritmo de clusterização chamado de *self-configurable clustering scheme* (SCCH). O SCCH agrupa os nós sensores e seleciona os CHs. Para definir os CHs é utilizado um sistema fuzzy onde 3 tipos de informações locais a cada nó sensor são consideradas como parâmetros de entrada, são eles: sua energia residual, a centralidade do nó que indica o quão central o nó está em relação a todos seus vizinhos da rede e a distância local que indica a soma das distâncias do nó até seus vizinhos dentro de um determinado raio de alcance. A saída do sistema *fuzzy* é um valor que representa a elegibilidade dos nós sensores quanto a se tornarem CHs. Então, os nós da rede comparam as suas elegibilidades entre si. Um nó com o valor máximo de elegibilidade vai apresentar-se como um CH, enquanto que o restante dos nós se apresenta como backup CHs (BCHs). Como resultado, os CMs podem se assegurar de que há sempre um BCH para os seus CHs. Portanto, em caso de falha permanente do CH, os CMs podem substituir o CH com o respectivo BCH. O CAMAW é diferente do SCCH na medida em que organiza a rede para atender múltiplas aplicações, enquanto o SCCH é uma abordagem de aplicação única. Ou seja, o CAMAW leva em consideração os requisitos de múltiplas aplicações no processo de escolha dos CHs. Já o SCCH busca uma escolha ótima de CHs que objetiva a cobertura da rede levando em consideração neste processo a distribuição geográfica dos nós pela rede e estabilidade dos CHs duarante a operação da rede.

Em Takavoli et al. (2010) é proposto o *Task-Cruncher*, um sistema voltado para RSCs que tira vantagem das eventuais redundâncias temporais/espaciais que podem existir entre múltiplas tarefas de sensoriamento em termos de seus respectivos fluxos de dados. Segundo o autor, o problema de minimização de redundâncias em múltiplos fluxos de dados tem sido tratado em otimização *multi-query* para bancos de dados. Já no *Task-Cruncher* este problema é modelado através de um grafo de intervalo de cobertura construído através de um algoritmo guloso a partir das “janelas de tempo” do sensoriamento de cada tarefa. Note que um grafo de intervalo de cobertura não é equivalente a um grafo de intervalo. Em um grafo de intervalo, dois vértices são vizinhos se e apenas se dois intervalos se sobrepõe uns com os outros em qualquer ponto, enquanto que no grafo de intervalo de cobertura a vizinhança entre dois vértices é determinada pela intersecção de um intervalo e o ponto final do outro intervalo. Já a

janela de tempo é composta pelo período definido para a tarefa, uma tolerância positiva e uma tolerância negativa, projetadas sobre um eixo temporal. O autor demonstra que o conjunto de vértices dominantes do grafo corresponde ao esquema ótimo de monitoramento. O autor argumenta que o sistema pode incrementar a escalabilidade da rede, pelo fato que é particularmente interessante aos chamados *hot sensors*, sensores com alta sobrecarga de trabalho por conta do interesse de diversos usuários e que , por conta desse maior volume de trabalho, podem se beneficiar em maior medida da economia de energia obtida com a eliminação de tarefas redundantes. Segundo o autor, embora o procedimento de detecção de redundâncias temporais entre fluxos de dados não seja trivial, ele argumenta que o custo computacional da detecção não supera a economia obtida com a otimização dos fluxos, mesmo nas situações de entrada e saída de aplicações. O CAMAW, de maneira similar mas computacionalmente mais simples que o *Task-Cruncher*, também busca a partir de uma linha do tempo eliminar eventuais redundâncias entre as aplicações, buscando coincidências em termos da periodicidade dos monitoramentos entre duas ou mais aplicações. O procedimento no CAMAW é mais simples porque ele precisa apenas da taxa do monitoramento de cada aplicação para buscar as sobreposições no eixo do tempo, enquanto que o TaskCruncher precisa de 3 informações de cada fluxo para poder estabelecer as sobreposições de monitoramento entre aplicações: a taxa, uma tolerância positiva e uma tolerância negativa. O objetivo é que o resultado do monitoramento da aplicação de maior periodicidade possa ser aproveitado sempre que houver sobreposições de período com aplicações de menor periodicidade. O *Task-Cruncher* assim como o CAMAW são soluções dinâmicas, pois ambos são capazes de lidar, em tempo de execução, com a entrada e saída de tarefas. Tanto o *Task-Cruncher* quanto o CAMAW utilizam a localização geográfica na definição do conjunto de sensores relevantes ao monitoramento e associam uma máscara que identifica o conjunto de aplicações atendidas por cada sensor. O *Task-Cruncher* apresenta alguns procedimentos centralizados como, o processo de seleção dos nós para a tarefa de monitoramento e o procedimento de construção do grafo a partir deste conjunto de nós. Segundo Afsar et al. (2014) e Younis et al. (2006) a abordagem centralizada em algoritmos de clusterização é um fator limitador para a escalabilidade da rede. O CAMAW, por outro lado, é um algoritmo completamente distribuído onde, uma vez que o nó sorvedouro anuncia a chegada ou saída de aplicações a todos os nós da rede, estes são capazes de trabalhar colaborativamente para se organizar em agrupamentos e realizar o monitoramento requerido, sem necessidade de auxílio externo (nó sorvedouro/estação base).

Em Gao et al.(2015) introduz-se o problema de compartilhamento de intervalo de dados das aplicações. A questão é como transmitir a menor quantidade de dados possível pela rede, de uma forma que estes dados satisfaçam aos requisitos de todas as aplicações. Diferentemente dos trabalhos atuais onde as aplicações em geral necessitam de amostras simples dos dados, o autor lida com aplicações que precisam de intervalo contínuo de dados. O autor argumenta que este é um problema de otimização não linear, não convexo e de grande complexidade e que, por isso, não é adequado a plataformas de sensores restritas em recursos de processamento. Para reduzir a complexidade do problema no âmbito das RSCs e com o objetivo de obter uma solução em tempo polinomial, o autor propõe um algoritmo guloso de aproximação com fator 2. O CAMAW também procura reduzir a quantidade de dados a transmitir com base na eliminação de redundâncias de monitoramento entre as aplicações; entretanto, ele o faz de uma forma simplificada pois se utiliza apenas a informação da taxa de monitoramento de cada aplicação para identificar as sobreposições no eixo do tempo. Enquanto que em Gao et. al (2015) o monitoramento das aplicações é modelado na forma de intervalos contínuos de amostragem para cada aplicação. Onde cada uma das amostras tem um início e um fim representando um intervalo no eixo do tempo. A sequência de intervalos associados a cada aplicação são analisados pelo algoritmo que busca uma redução da quantidade total de amostras por compartilhar a amostragem realizada entre as múltiplas aplicações.

Finalmente, em Farias et al. (2013) é proposto um algoritmo de escalonamento voltado para as RSCs que explora características comuns entre as aplicações como um meio para melhorar a eficiência do sistema. Para atingir este objetivo o autor define as aplicações como sendo compostas por tarefas de monitoramento, e as modela na forma de DAGs (*Directed Acyclic Graphs*). Sendo assim, cada aplicação é representada por um DAG e cada nó de um DAG representa uma tarefa da aplicação. Podem fazer parte de um DAG tarefas periódicas e não periódicas. As arestas representam as dependências entre tarefas. Os DAGs com tarefas em comum podem ser combinados e esta composição pode então ser alocada em uma RSC. O CAMAW também procura, com base nos requisitos das aplicações, identificar as redundâncias de monitoramento como forma de evitar a execução repetida de tarefas comuns a mais de uma aplicação. Entretanto o faz uma forma distribuída e portanto mais escalável que o trabalho de Farias et al. (2013) cuja abordagem é centralizada para a geração e combinação dos grafos.

3.1 Considerações sobre os trabalhos relacionados

Neste capítulo foram apresentados trabalhos que apresentam similaridades com o CAMAW como, por exemplo, soluções para o problema do compartilhamento dos recursos das RSSFs entre múltiplas aplicações. Estes trabalhos, assim como o CAMAW, também utilizam as técnicas de **clusterização** e/ou **compartilhamento de dados**, tanto para suportar o monitoramento de múltiplas aplicações quanto para incrementar a eficiência energética da rede através da eliminação de tarefas redundantes. As redundâncias podem ocorrer nas tarefas de sensoriamento, armazenamento, transmissão dos dados ou escolha dos CHs da rede. Os trabalhos que lidam com múltiplas aplicações podem ou não considerar **requisitos da aplicação** para a escolha do conjunto de nós responsável pelo monitoramento.

Tabela 1. Comparação dos trabalhos relacionados

Trabalhos	Clusterização	Requisitos de aplicação	Compartilhamento de Dados	Múltiplas Aplicações	Multi-Hop	Distribuído
CAMAW	Sim	Sim	Sim	Sim	Sim	Sim
S-LEACH	Sim	Não	Sim	Sim	Sim	Sim
SCCH	Sim	Não	Não	Não	Não	Sim
SHAAL	Sim	Sim	Não	Sim	---	Não
SENSOMAX	Sim	Sim	Não	Sim	Sim	Sim
SENSHARE	Sim	Sim	Não	Sim	Sim	Não
Gao et al.(2015)	Não	Não	Sim	Sim	---	---
Task-Cruncher	Não	Não	Sim	Sim	---	Não
Farias et al.(2013)	Não	Sim	Sim	Sim	Sim	Não

4 Descrição do CAMAW

O CAMAW é um algoritmo de clusterização hierárquico, distribuído e voltado para as redes de sensores compartilhadas. Tradicionalmente, os algoritmos de clusterização criam agrupamentos compostos por nós CM e nós CH, em uma estrutura hierárquica que é considerada uma solução escalável e de baixo consumo energético, tanto para gerir a organização da rede, quanto para a realização das tarefas de coleta e transmissão de informações ambientais. O CAMAW também se enquadra nesta descrição, embora seus procedimentos e estruturas de dados tenham sido desenhados para lidar com as complexidades inerentes à chegada, saída e execução de múltiplas aplicações concorrentes. Além disso, ele procura tirar proveito de um ambiente com múltiplas aplicações para eliminar redundâncias nas tarefas de coleta e transmissão dos dados como uma forma de economizar energia.

No CAMAW, cada período de execução do algoritmo é considerado um ciclo. Cada ciclo inicia-se pela sincronização de todos os nós na RSSF e, para este processo, podemos utilizar um dos algoritmos de sincronização disponíveis na literatura como o apresentado em Maroti et al. (2004). Após a sincronização, os nós devem esperar por mensagens provenientes do nó sorvedouro que indiquem a criação de uma nova aplicação, o CAMAW é responsável por clusterizar os nós para as aplicações seguindo as capacidades dos nós e os requisitos das aplicações. Caso contrário, se o tipo da mensagem indica o encerramento de uma aplicação, dois casos são possíveis: em primeiro lugar, se a aplicação é a única aplicação no *cluster*, os nós devem manter a formação do *cluster*, mas parar todas as atividades de monitoramento; em segundo lugar, se há outras aplicações na rede, os nós devem liberar os recursos usados por esta aplicação no *cluster*, mantendo as atividades de monitoramento.

O CAMAW está apenas preocupado com a formação dos *clusters* e a coleta e transmissão dos dados ambientais. Outros procedimentos, como o roteamento dos dados dos sensores até o nó sorvedouro e a realização de procedimentos de fusão de dados estão fora do escopo do presente trabalho.

4.1 Visão Geral do algoritmo CAMAW

O CAMAW é composto por três fases: (i) Setup (descrição detalhada em seção 4.3), que é responsável por configurar os parâmetros iniciais do algoritmo; (ii) Chegada de aplicação (descrição detalhada na seção 4.4), que é responsável pela organização da rede em grupos de acordo com as capacidades dos nós e com os requisitos de monitoramento das aplicações; e (iii) Saída de aplicação (descrição detalhada na seção 4.6), que é responsável por refletir nos

agrupamentos da rede a saída de uma aplicação. Este procedimento somente ocorre se a existência dos agrupamentos vigentes depender exclusivamente da aplicação terminada. O pseudocódigo do CAMAW pode ser visto na Figura 1.

```
Entradas: Aplicações que são instaladas na rede (AppRequirements), NodeCapabilities
Saídas: Clusters por aplicação
1.  # FASE DE SETUP
2.  Preencher NodeCapabilities
3.  Para cada novo round
4.          Executar um algoritmo de sincronização
5.          Se não é a primeira rodada
6.                Para cada aplicação j
7.                          ROLE_SELECTION_PROCEDURE()
8.                          ASSOCIATION_PROCEDURE()
9.  Espera por mensagens
10. Se mensagem = BS_NEW_APP
11.             Para cada Aplicação A_i em A
12.                     APPLICATION ARRIVAL PHASE ()
13. Senão se messagem = BS_END_APP ou Se (papel do nó = CH e tDur expirado)
14.                     APPLICATION DEPARTURE PHASE ()
```

Figura 1. Procedimento de formação de clusters do CAMAW

4.2 Estruturas de dados

A rede é constituída por um conjunto *V* de nós sensores $v_i \in V$, onde $V = \{v_1, v_2, ..., v_n\}$, na qual podem estar ativas um conjunto de aplicações $a_j \in A$, onde $A = \{a_1, a_2, ..., a_m\}$. Cada nó *vi* pertencente a *V* pode atender de 0 até o total de aplicações pertencentes a *A*, dependendo da aptidão do nó para realizar o monitoramento requerido por cada aplicação. Durante a execução do algoritmo existem dois estados possíveis para as aplicações na rede: ativa ou inativa. Uma aplicação está ativa se houver ao menos um agrupamento de nós associado a esta aplicação. Por outro lado, uma aplicação estará inativa sempre que não houver agrupamentos associados à aplicação.

As estruturas de dados usadas pelo CAMAW são: *NodeCapabilities* e *AppRequirements*. A estrutura *NodeCapabilities* contém os seguintes campos: um identificador único de nó sensor, tal como o endereço MAC do nó; o *NodeID;* os tipos de dispositivos de sensoriamento existentes no nó, em *TpMnt;* e as respectivas taxas de coleta de dados em uso, o *TxUse.*

A estrutura *AppRequirements* pode ser instanciada várias vezes, uma vez para cada aplicação. Ela contém o campo de identificador da aplicação (*AppID*) em estado ativo. Além disso, essa estrutura contém os seguintes campos: (i) as coordenadas que definem a localização geográfica e a área de interesse da aplicação, representada pelos atributos *coordX1*, *coordY1*, *coordX2* e *coordY2*; (ii) *Aptitude*, informação que revela se o nó é capaz de monitorar para uma determinada aplicação (0 = não apto e 1 = apto); (iii) *TDur*, tempo em que a aplicação deve permanecer em execução na rede, ou seja, a duração da aplicação; (iv) *TpMnt*, requisitos de monitoramento, como as unidades de sensoriamento e as respectivas taxas de sensoriamento, a *TxApp*; (v) *IsCollector* indica o papel do nó, se CM ou CH para a aplicação; e (vi) *nodeID*, o identificador do nó cujo papel é CH para a aplicação; (vii) identificadores de todos os nós vizinhos capazes de monitorar para esta aplicação, na estrutura *NeighborSet*. Além disso, para cada nó vizinho, é armazenado o atributo *nodeUtility* que informa o quão promissor (informação obtida realizando o cálculo da função W descrita no item 4.4.2) o nó é para se tornar CH para dada aplicação. A Figura 2 exibe o diagrama de classes do protótipo do CAMAW que exibe em suas classes as estruturas definidas nesta seção.

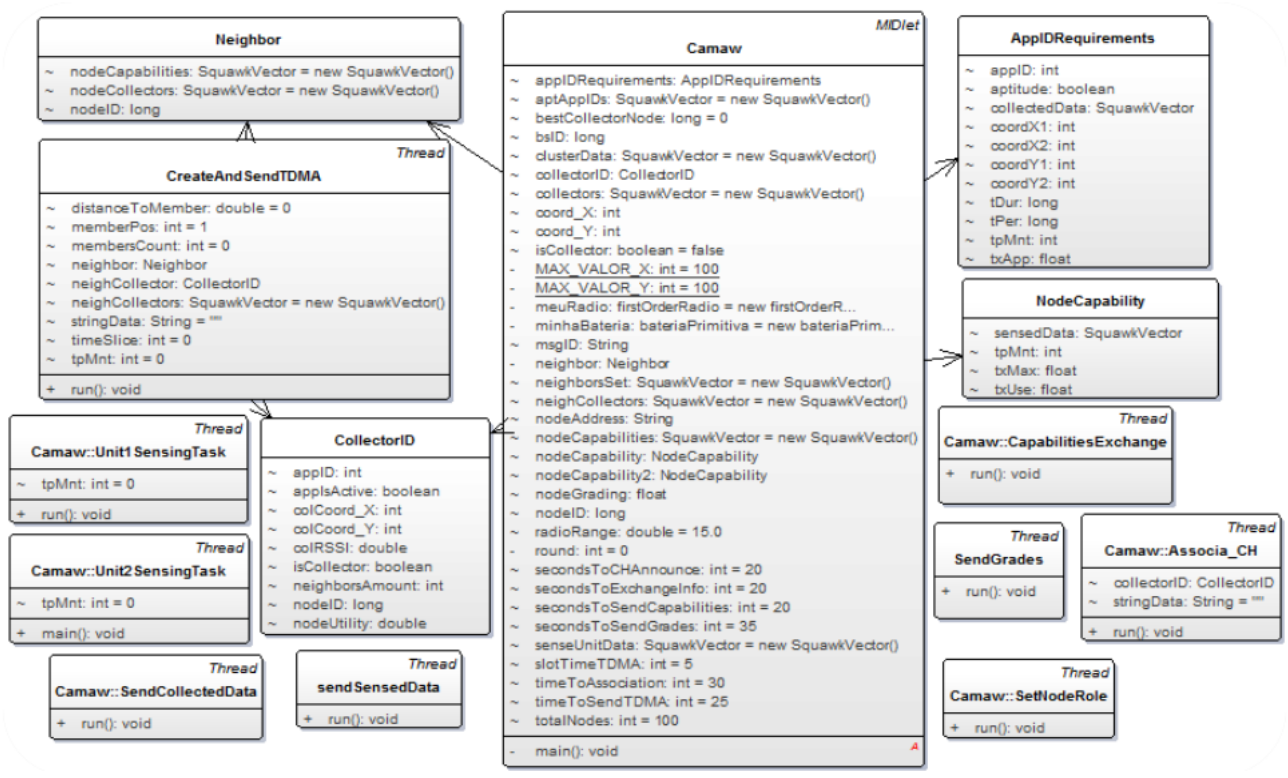


Figura 2. Diagrama de classes do CAMAW

É importante mencionar que um nó pode ser considerado apto ou não apto. Para ser considerado apto deve possuir uma ou mais unidades de sensoriamento requeridas pela aplicação e estar localizado na área de interesse da aplicação. Foi introduzida uma função de aptidão que indica se um sensor pode fornecer o serviço exigido pela aplicação. A função está descrita pela Eq. (1), onde t é a unidade de sensoriamento requerida por uma aplicação, x_1,y_1,x_2 e y_2 são coordenadas da localização geográfica para o monitoramento de eventos e i é o *NodeID* do sensor.

$$A(t,x_1,y_1,x_2,y_2,i) = \begin{cases} 1, & se\ um\ sensor\ está\ apto \\ 0, & se\ um\ sensor\ está\ não\ apto \end{cases} \quad (1)$$

4.3 Fase de Setup

Esta fase é responsável por configurar os nós sensores v_i e popular as estruturas de dados que são necessárias durante as demais fases do algoritmo. Durante a fase de setup é executado um procedimento de sincronização (MAROTI et al., 2004) de forma a garantir a correlação espacial e temporal dos dados coletados pela RSSF. A sincronização torna possível que o algoritmo inicie a aquisição de dados simultaneamente em vários nós. Além disso, no inicio de cada ciclo, cada nó v_i executa o procedimento de seleção de papel para cada aplicação a_j, denominado de ROLE_SELECTION_PROCEDURE, detalhado na subseção 4.4.2. A rotação de papéis é utilizada para amenizar o esgotamento da energia dos nós CH e uniformizar a dissipação de energia entre os nós da rede.

4.4 Fase de chegada de aplicação

Esta fase é a responsável por organizar os nós da rede nos agrupamentos que irão atender as necessidades de monitoramento das aplicações. Para atingir este objetivo são consideradas neste processo, as capacidades de monitoramento dos nós e os requisitos de monitoramento das aplicações. Esta fase é subdividida nos seguintes quatro procedimentos: **Verificação de Aptidão**, **Seleção de Papel**, **Associação** e **Coleta de Dados**. No procedimento de Verificação de Aptidão, o nó verifica se ele está apto a monitorar para a nova aplicação. No procedimento de seleção de papel, cada nó apto decide o seu papel para a nova aplicação entre CH ou CM. No procedimento de Associação, cada nó CM é responsável por se associar com o seu respectivo CH ou, caso seja CH, deve esperar pelo envio de solicitações de associação de nós CM.

4.4.1 Procedimento de Verificação de Aptidão

O objetivo desse procedimento é determinar se o nó sensor i é capaz de atender aos requisitos de monitoramento de determinada aplicação. Neste procedimento, cada nó sensor aguarda receber a mensagem BS_NEW_APP a partir do nó sorvedouro. A mensagem BS_NEW_APP contém os parâmetros de monitoramento requeridos pela aplicação que chega à rede, entre eles: a identificação das unidades de monitoramento como, por exemplo, temperatura e luminosidade, que ficam armazenados na estrutura de dados *AppRequirements.TpMnt* do nó sensor*;* suas respectivas taxas, armazenadas em *AppRequirements.TxApp;* as coordenadas da área de interesse da aplicação armazenadas nos atributos *coordX1*, *coordY1*, *coordX2* e *coordY2* da estrutura *AppRequirements* e por fim, a duração do monitoramento da aplicação, armazenada em *AppRequirements.TDur*. De posse dessas informações o nó sensor realiza duas verificações: primeiro, se contém ao menos uma das unidades de sensoriamento requeridas; segundo, se está dentro da área de monitoramento de interesse da aplicação. Caso as verificações retornem positivas, o nó sensor será atualizado com a nova aplicação em suas estruturas de dados, entre elas o identificador de aplicação *AppRequirements.AppID*, sua aptidão *AppRequirements.AppID.Aptitude=True* e os requisitos de monitoramento da nova aplicação *AppRequirements.TpMnt, AppRequirements.TxApp, AppRequirements.TDur* e por fim, as coordenadas geográficas (*coordX1*, *coordY1*, *coordX2* e *coordY2*) em *AppRequirements*. Caso a verificação retorne negativa, o nó aguarda em baixo ciclo de trabalho de forma a poupar sua energia até que surjam novas aplicações. A partir deste ponto e somente para os nós que forem aptos é dado início ao procedimento seguinte, de seleção de papel.

Todo o procedimento descrito acima realiza a avaliação de aptidão do nó para uma única aplicação. Entretanto, para situações em que o nó se encontra em uma área de interesse de mais de uma aplicação, o procedimento de clusterização deverá abranger a totalidade de aplicações interessadas na região geográfica onde o nó se encontra. Para ilustrar esta situação na Figura 3 aparecem retângulos que representam as áreas de interesse das aplicações 1, 2 e 3 para o monitoramento da temperatura. Se o nó estiver em qualquer região hachurada, significa que está na área de interesse de mais de uma aplicação. Caso o nó esteja na região hachurada em cinza claro entre as aplicações 1 e 2, o identificador da aplicação (*AppRequirements.AppID*) será “12”, de forma a representar as duas aplicações em conjunto. Esta representação além de indicar o monitoramento para mais de uma aplicação também identifica as aplicações envolvidas. Tal representação é similar ao *bitmask* utilizado para

identificar e organizar o monitoramento das aplicações em Takavoli et al. (2010). Em relação à taxa de monitoramento, é calculada uma taxa de maneira a atender ambas aplicações, a menos que uma taxa seja múltipla em relação a outra. Nesse caso, utiliza-se a taxa mais exigente (em frequência de monitoramento) entre as aplicações, pois irá atender a ambas. A definição da taxa está relacionada ao procedimento de coleta de dados e está descrito na subseção 4.4.4. A clusterização de nós localizados em áreas que interessam a mais de uma aplicação é personalizada e acontece separadamente das demais regiões da rede. Nas situações em que o nó está na região de interesse de três ou mais aplicações, como a região hachurada em cinza escuro de interesse das aplicações 1, 2 e 3, embora todos os nós desta região possam atender às combinações {1,2,3,12,13,123} de aplicações, o *cluster* formado nesta região compreende todas as aplicações interessadas, ou seja 1,2 e 3 e será composto apenas por nós aptos às três aplicações. O identificador da aplicação (*AppRequirements.AppID*) neste caso será “123”. Este é um arranjo mais eficiente, pois limita o procedimento de clusterização à àrea de interesse das aplicações, criando *clusters* capazes de atender ao conjunto de aplicações.

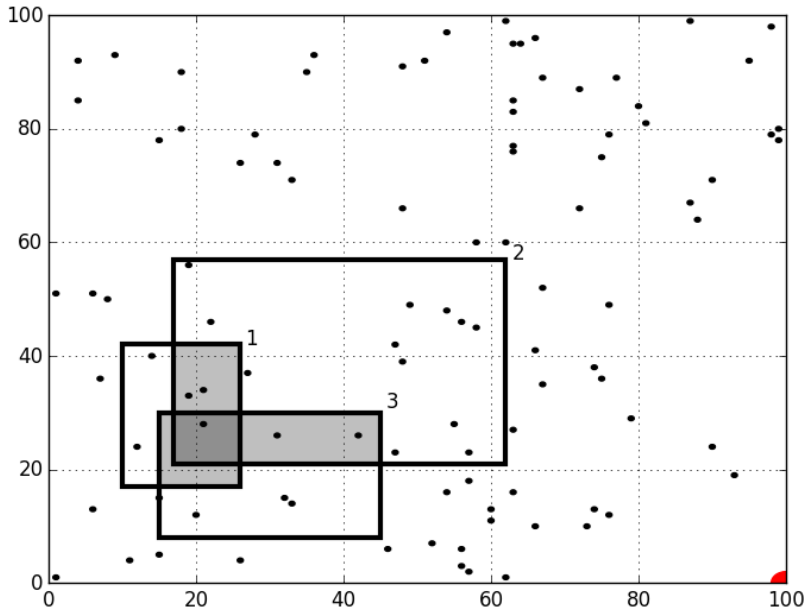


Figura 3. Aptidão com base na localização geográfica

4.4.2 Procedimento de seleção de papel

O objetivo desse procedimento é determinar o papel de cada nó sensor v_i para as novas aplicações a_j, de acordo com uma função de utilidade W_i. Primeiro é apresentada a função utilidade que informa "o quão promissor" é um dado nó sensor v_i da rede para assumir o papel

de líder de *cluster* (CH) para a nova aplicação a_j. Em seguida, o próprio processo de seleção de papel é descrito.

Função Utilidade

Wij é calculado para medir a utilidade de um dado nó sensor v_i para a nova aplicação a_j como uma função do (i) nível de energia residual do nó sensor v_i e (ii) a percentagem de nós vizinhos dentro do alcance do rádio do nó sensor v_i. A função de utilidade *W(i,j)* é apresentada na Equação 2.

$$W_{(i,j)} = X_{ij} + Y_i \qquad (2)$$

Na equação 2, *Xij* indica o percentual de nós da rede que são vizinhos ao nó v_i, considerando a nova aplicação a_j, e *Yi* informa a energia residual do nó v_i. *Xij* é definido (Equação 3) como a razão entre o número de vizinhos do nó v_i para a nova aplicação a_j, e a quantidade total de nós da RSSF, representada por *N*. Ou seja, o número de vizinhos do nó é normalizada (dividida) pelo total de nós da rede, de modo que o valor mais alto de *Xij* é igual a 1, indicando a situação em que todos os nós da rede são vizinhos, ao alcance do rádio, do nó atual. A energia residual é definida (Equação 4) como a quantidade atual, em termos percentuais, de energia remanescente no nó sensor v_i. Para se obter *Yi* é realizado a aferição do nível de energia do nó e este resultado é normalizado (dividido) pela quantidade energia máxima alcançada por este nó, de modo que quando a bateria do nó está completamente carregada este valor corresponde a 1.

$$Xi,j = \frac{\sum Vizinhosij}{N} \qquad (3)$$

$$Yi = \frac{Ei\ residual}{Ei\ total} \qquad (4)$$

Seleção de papel

O objetivo desse procedimento consiste em selecionar o papel apropriado do nó v_i. Neste procedimento (Figura 4), o nó sensor apto v_i calcula a sua utilidade pela função *Wij* (equação 2). Após obter o valor de sua utilidade para a aplicação a_j, o nó sensor v_i armazena esta informação no atributo *NodeUtility*. Em seguida o nó sensor v_i envia a seus vizinhos a mensagem CAPABILITIES_EXCHANGE (linha 4 da Figura 4), contendo a sua utilidade para a aplicação a_j e seus recursos (*NodeCapabilities*).

O nó sensor v_i aguarda para receber as mensagens CAPABILITIES_EXCHANGE de seus vizinhos referentes à chegada de uma nova aplicação a_j. Para cada mensagem CAPABILITIES_EXCHANGE recebida e para cada aplicação a_j, o nó sensor v_i atualiza sua estrutura *NeighborSet* com os identificadores dos seus nós vizinhos (*NodeCapabilities.NodeID*), e seus respectivos valores de utilidade (linha 8 da Figura 4). Além disso, também são atualizados os tipos de unidades de monitoramento (*TpMnt*) que estão presentes em cada nó vizinho.

Com esta informação, cada nó sensor v_i compara seu valor de utilidade em relação aos seus vizinhos. Para cada aplicação a_j, o nó sensor v_i verifica se contém o maior valor de utilidade e também se possui nós em sua vizinhança que possam atender o monitoramento requerido pela aplicação. Caso positivo, o nó irá enviar a mensagem NEW_COLLECTOR para os seus vizinhos, a fim de informá-los de que é o novo CH da aplicação a_j naquela região (linhas 9-13 da Figura 4). A mensagem NEW_COLLECTOR contém o identificador do CH (*NodeCapabilities.NodeID*). Para cada aplicação a_j, os vizinhos que receberam a mensagem NEW_COLLECTOR se tornarão CMs (linha 13 da Figura 4) para a aplicação a_j.

No caso da chegada de uma nova aplicação a_j', o nó v_i, que já atua como CH para a aplicação a_j, verifica com o auxílio da sua estrutura *AppRequirements*, se também pode ser CH para a nova aplicação a_j'. Ele irá verificar se o conjunto de CMs em *AppRequirements.NeighborSet* de a_j é igual ao conjunto necessário à a_j' (linha 6 da Figura 4). Se todos os CMs em *AppRequirements.NeighborSet* estão aptos para monitorar para a nova aplicação, então o CH atualiza sua *AppRequirements.TxApp* com uma taxa de monitoramento que atenda a ambas aplicações e adiciona o *AppID* dessa nova aplicação à sua estrutura *AppRequirments* (linha 24-25 da Figura 4). Senão, no caso apenas alguns nós em *AppRequirements.NeighborSet* serem aptos ao monitoramento requerido pela aplicação a_j, o nó v_i irá enviar uma mensagem CH_END_CLUSTER (linha 26 da Figura 4) para os nós. Em seguida, o nó v_i irá enviar a mensagem CH_NEW_APP (linha 27 da Figura 4) para os nós realizarem novos procedimentos de seleção de papel e de associação, com objetivo de escolher um CH mais adequado ao monitoramento requerido pelo conjunto de aplicações. A Figura 4 apresenta o pseudocódigo do procedimento de seleção de papel.

Entradas: Requisitos da aplicação *(AppRequirements)*, Características do nó apto (*NodeCapabilities*)

Saídas: papel do nó (CM/CH).

```
1.  #PROCEDIMENTO DE SELECAO DE PAPEL
2.      Se nó está apto E não possui papel
3.          Define classificação do nó através da equação (2)
4.          Envia msgs CAPABILITIES_EXCHANGE aos vizinhos
5.          Espera por msgs CAPABILITIES_EXCHANGE da vizinhança, durante uma fração
    do slot time da fase de setup de uma rodada
6.          Armazena capacidades dos vizinhos, obtidas a partir de mensagens
    recebidas, na estrutura de dados AppRequirements.NeighborSet do nó.
7.          Para cada nó vizinho <i> em AppRequirements. NeighborSet
8.           Se melhorClassificação <= i classificação
9.                Muda melhorClassificação para i classificação
10.          Se      ClassificaçãoNo      em      AppRequirements.NeighborSet      >
    melhorClassificação
11.              Envia NEW_COLLECTOR para todos os nós vizinhos
12.          Senão
13.              Espera para todas NEW_COLLECTOR durante uma fração do slot time da
    fase de setup de uma rodada
14.               Atualiza     capacidades     do     candidato     a     CH     em
    AppRequirements.NeighborSet a partir das msgs recebidas
15.          Se nó já tem o papel de CH
16.              Para cada nó monitor i em AppRequirements.NeighborSet
17.                  Para cada TpMnt de i em AppRequirements
18.                    Para cada TpMnt de cada novo AppID em AppRequirements
19.                                  Se TpMnt da nova AppID em AppRequirements
    corresponde ao TpMnt de i em AppRequirements.NeighborSet
20.                               Adiciona i em newClusterStructure
21.                               Se TpMnt de AppID em AppRequirements não existe em
    AppRequirements.AppID
22.                                                Armazena TpMnt de AppID em
    AppRequirements.AppID
23.                               Senão
24.                                        Se AppRequirements .TxApp de AppID >
    AppRequiremnts.TxApp de i
25.              Atualiza AppRequirements.TxApp de i com AppID.TxApp
26.              Se newNeighborSet é igual a Neighbors em AppRequirements
                          Envia CH_END_CLUSTER para nós newNeighborSet
27.                      Envia CH_NEW_APP  para nós newNeighborSet
28.          Senão
29.               Envia UPDATE_SENSORING com novas NeighborSet in AppRequirements
    configurações para todos os nós em AppRequirements
```

Figura 4. Procedimento de seleção de papel

4.4.3 Procedimento de Associação

Para cada nova aplicação *aj*, o nó sensor v_i verifica o seu papel. Se o papel nó v_i é CM, este nó escolhe um nó CH ao alcance do seu rádio para se associar, entre os nós CHs da região em que está localizado, escolhendo o CH com maior força de sinal, ou seja, o CH com o maior valor de *Received Signal Strength Indicator* (RSSI). Depois de escolher o nó CH, o nó CM envia uma mensagem JOIN_CLUSTER a ele. Essa mensagem contém o identificador do nó (*NodeCapabilities.NodeID*) e o identificador da nova aplicação *aj*. Em seguida, o nó CM v_i aguarda para receber a mensagem UPDATE_SENSORING de seu nó CH, informando que o nó pode começar a coletar dados para a nova aplicação *aj*. Esta mensagem contém o identificador da nova aplicação, os tipos de monitoramento (*AppRequirements.TpMnt*) e suas respectivas taxas (*AppRequirements.TxApp*). Com esta informação sobre a nova aplicação *aj*, o nó CM v_i atualiza seu atributo *NodeCapabilities.TxUse*. A mensagem também define a participação do nó no ciclo TDMA de comunicação *intra-cluster*.

Se o papel escolhido para o nó v_i for CH, então o nó aguarda receber a mensagem CM_JOIN de nós CM que serão membros do novo *cluster* para a aplicação *aj*. Depois de receber cada mensagem CM_JOIN, o nó CH v_i atualiza em *AppRequirements* as entradas referentes a cada nó CM responsáveis pelos envios das mensagens CM_JOIN. Em seguida, o nó CH v_i envia uma mensagem UPDATE_SENSORING para seus nós CMs. A Figura 5 apresenta o pseudocódigo do procedimento de associação.

Entradas: Aplicações que são instaladas na rede *(AppRequirements)*, *NodeCapabilities*

Saídas: Nó associado a um cluster (CH_ID ≠ Null).

```
1.  # PROCEDIMENTO DE ASSOCIAÇÃO
2.  Se papel do nó = CM
3.        Escolhe o CH com maior RSSI em AppRequirements
4.        Envia o CM_JOIN para o CH escolhido
5.        Atualiza seu CH_ID em AppRequirements com o nodeID do CH escolhido.
6.  Senão se papel do nó = CH
7.        Espera por todos os CM_JOIN dos nós vizinhos.
8.        Com os dados contidos nas mensagens recebidas dos nós da vizinhança,
    atualiza a entrada correspondente ao nó em NeighborSet como nó monitor.
9.         Envia UPDATE_SENSORING para esses nós presentes em AppRequirements
10.        Atualiza seu CH_ID com o nodeID do nó sorvedouro.
```

Figura 5. Procedimento de associação

4.4.4 Procedimento de coleta de dados

O objetivo deste procedimento é coletar e entregar os dados ambientais obtidos da RSSF às aplicações de uma forma compartilhada e energeticamente eficiente. A partir do

término do procedimento de associação, a rede está completamente organizada, com cada nó apto ciente do seu papel em seu *cluster*. Assim, pode ter início o procedimento de coleta de dados. Durante este procedimento os nós trabalham de forma sincronizada na realização das tarefas de coleta, agregação e transmissão dos dados. Resumidamente, o procedimento de coleta de dados nos algoritmos de clusterização é composto pela seguinte sequência de eventos: os nós membros do *cluster* coletam dados ambientais e os enviam para seu respectivo líder que por sua vez os remete em direção ao nó sorvedouro (capítulo de conceitos básicos). No CAMAW, este procedimento é um ponto chave na busca por eficiência energética, uma vez que identificadas redundâncias acerca do monitoramento requerido por cada aplicação, é possível eliminá-las e reduzir em consequência disso os esforços de monitoramento e transmissão de dados pela rede. Quando diferentes aplicações tem um interesse em uma mesma unidade de sensoriamento procura-se estabelecer um esquema de sensoriamento que é resultado da combinação das taxas requeridas por cada aplicação. O exemplo da Figura 6 exibe um cenário com 3 aplicações (A,B e C) que demandam dados de monitoramento de uma mesma unidade como, por exemplo, temperatura, mas com diferentes taxas de amostragem. São identificadas oportunidades de redução no número de coletas nos períodos S1, S2, S4, S6, S7 e S8. Sendo que no instante S4 está a maior oportunidade de economia, pelo fato de que as três aplicações podem ser atendidas com apenas uma coleta.

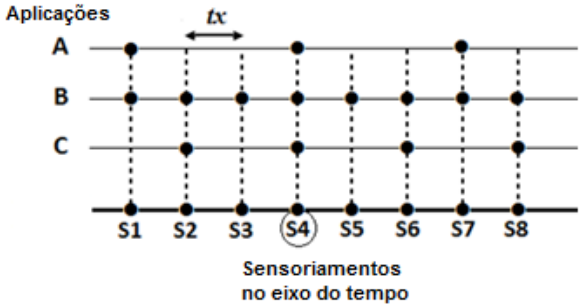


Figura 6. Redundâncias de monitoramento entre as aplicações

A transmissão dos dados é realizada seguindo a mesma lógica de eliminação de redundâncias observada no procedimento de coleta de dados. Na Figura 7, os dados coletados em S4 são transmitidos uma única vez, mas endereçando as três aplicações ao mesmo tempo a partir do mesmo pacote de dados. Ou seja, o pacote carrega em seu "*payload*" além do dado coletado, identificadores para cada uma das aplicações com interesse naquele dado. Cabe ao nó sorvedouro a leitura destas informações e a entrega às aplicações interessadas do dado contido no pacote. Nas abordagens tradicionais, como o Senshare, o SHAAL, o Sensomax, entre outros, cada aplicação executa em uma instância própria e isolada da rede, o que

possibilita a ocorrência de tarefas redundantes de coleta e transmissão de dados em um ambiente com múltiplas aplicações.

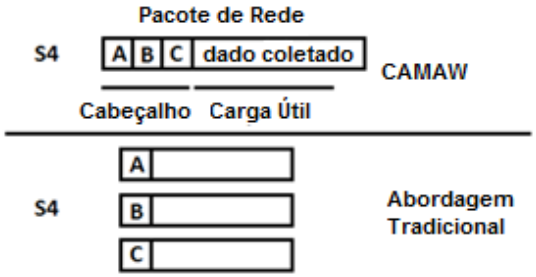


Figura 7. Transmissão dos dados no CAMAW e nas abordagens tradicionais

Ciclo de operação do algoritmo

O CAMAW é um algoritmo que cria os agrupamentos pela rede em resposta aos eventos de entrada e saída de aplicações, mas que também repete estes procedimentos de forma periódica como os algoritmos LEACH, SCCH, entre outros. Para possibilitar a clusterização periódica da rede foi definido um ciclo de operação, onde alguns procedimentos da fase de chegada de aplicação são repetidos em um intervalo fixo de tempo. A Figura 8 exibe o diagrama de atividades da fase de chegada de aplicação e também ilustra o ciclo de operação do algoritmo. Neste diagrama podem ser observados o fluxo das atividades relacionadas aos procedimentos de aptidão, seleção de papel associação e coleta de dados. Também é possível verificar os papéis que os nós podem assumir de acordo com cada fluxo de atividades seguidos pelos nós.

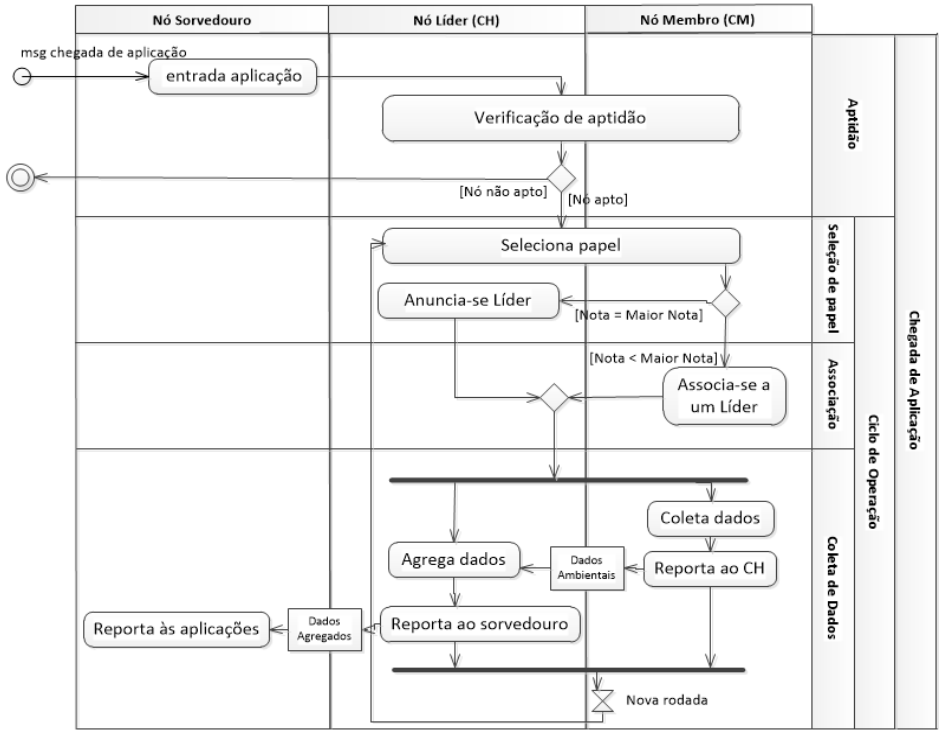


Figura 8. Diagrama de atividades com a fase de chegada de aplicação

4.5 Fase de saída de aplicação

Nesse procedimento, cada nó sensor v_i espera para receber a mensagem BS_END_APP enviada a partir do nó sorvedouro, ou espera que o tempo de duração de aplicação definido (*AppRequirements.TDur*) tenha terminado. O pseudocódigo dessa fase pode ser visto na Figura 9.

Entradas: Todos os nós com papéis definidos

Saídas: Nós livres em modo de espera

```
    # EXPIRACAO DA DURACAO DA APLICACAO
1.          Para cada AppID em AppRequirements
2.              Para cada TpDur  em AppRequirements.AppID
3.                  Se TpDur expirates
4.                      remover AppRequirements.AppID
5.                  Senão se MaxRate é Nulo OU (MaxRate.TpMnt = AppID.TpMnt E
    MaxRate.TxApp < AppID.TxApp)
6.                      MaxRate= AppID
7.          Se AppRequirements é Nulo
8.              Modifica todos NodeCapabilities.TxUse = 0
9.              removeCluster = true
10.         Senão
11.             Para cada TpMnt em NodeCapabilities
12.                 Para cada TpMnt em MaxRate
13.                     Se NodeCapabilities.TpMnt = MaxRate .TpMnt
14.                         NodeCapabilities.TxUse = MaxRate .TxApp
```

Figura 9. Fase de saída de aplicação

Para cada aplicação a_j, o nó sensor v_i verifica o seu papel. Se o papel do nó v_i é CM, ele espera receber a mensagem GO_TO_SLEEP enviada a partir do nó CH (linha 2). Esta mensagem vai parar as tarefas de monitoramento de uma aplicação (*Nodecapabilities.TxUse* receberá 0). Essa mensagem contém o *AppRequirements.AppID* das aplicações que estão deixando a RSSF. Se o nó monitora para uma única aplicação, ele passa ao estado de espera (*sleep mode*). Senão, o nó encerra o monitoramento para esta aplicação, mas ele mantém o monitoramento para as outras aplicações.

Se o papel do nó v_i é CH, existem duas possibilidades. Em primeiro lugar, se o nó é CH para uma única aplicação (linha 9) (há apenas um *AppID* em *AppRequirements*), a aplicação não é encerrada, para evitar um novo procedimento de clusterização. Nesse caso é preservada a estrutura do *cluster*, mas sem a coleta de tarefas ou transmissão de dados (ajustando *NodeCapabilities.TxUse* para 0) (linha 14-15) e em seguida os nós entram em um estado de baixo ciclo de trabalho, até que a aplicação retorne, ou até que apareça uma nova aplicação com interesse na mesma região. Caso contrário, se o nó tem o papel de CH para mais de uma aplicação, é feita uma busca por aplicações (em *AppRequirements.ApppID*) com características de monitoramento idênticas (*AppRequirements.TpMnt*) (linha 13) a da aplicação que sai da rede. Se não houver nenhuma outra aplicação que monitora para o mesmo tipo de monitoramento, o nó v_i envia a mensagem UPDATE_SENSORING (contendo *AppRequirements.AppID*) para todos os CMs no *cluster* dessa aplicação. Caso contrário, se há mais aplicações com mesmas características de monitoramento (quanto a *AppRequirements.TpMnt*), existem duas possibilidades. Primeiro, se a aplicação que está deixando o *cluster* tem a taxa de monitoramento mais exigente (*AppRequirements.TxApp*), então o nó v_i irá atualizar a taxa de monitoramento (*AppRequirements.TxApp*) para essa interface de monitoramento (*AppRequirements.TpMnt*) (linha 15), utilizando a taxa de transmissão da aplicação que permanece no *cluster*. Em seguida, o nó envia uma mensagem UPDATE_SENSORING (contendo o *AppRequirements.AppID*, *AppRequirements.TxApp*, *AppRequirmments.TpMnt*) para todos CMs. Segundo, se a aplicação que está deixando o *cluster* tem uma taxa de monitoramento (*AppRequirments.TxApp*) menos exigente do que a aplicação que parte, pode não ser necessário modificar a taxa de monitoramento (*AppRequirments.TxApp*). Neste caso, o nó v_i envia uma mensagem GO_TO_SLEEP contendo o *AppRequirements.AppID* da aplicação que parte para todos os CMs. Os CMs irão então parar de monitorar para esta aplicação.

4.6 Tolerância à falha

Com o objetivo de detectar e recuperar eventuais falhas ou mau funcionamento de nós CH nos agrupamentos da rede e assim evitar a perda continuada de informações de monitoramento, foi definido um procedimento de recuperação de falha. Este procedimento se inicia quando os nós de determinado agrupamento detectam que o seu respectivo CH completou o segundo ciclo TDMA sem utilizar o seu *slot-time* para se comunicar. Dessa forma, cada nó do *cluster* é capaz de perceber a falha do CH e em seguida dar início a um procedimento reduzido de clusterização, restrito à área de atuação do CH defeituoso para proceder a escolha de um novo líder. Não é necessário realizar todos os procedimentos descritos na seção 4.4 para restaurar o *cluster*. A clusterização pode ser dar a partir do procedimento de seleção de papel descrito na subseção 4.4.2, pois é realizado um reaproveitamento da nota de cada sensor a partir do procedimento de aptidão ocorrido na última rodada. Deste ponto em diante segue-se o fluxo normal do algoritmo com o procedimento de associação encerrando a clusterização. É importante frisar que no decorrer dessa nova eleição de líder os nós permanecem realizando em paralelo suas tarefas de monitoramento. Os dados coletados ficam sendo armazenados continuamente em um buffer modelado para suportar o período de escolha de um novo CH. Isso tem o objetivo evitar lacunas de monitoramento e garantir o fornecimento de uma amostragem contínua de dados para as aplicações. A partir do momento que estiver definido o novo líder ou líderes, os nós membros descarregam o buffer e encaminham os dados para o novo líder, restabelecendo a coleta de dados do ponto em que a falha ocorreu em diante.

5 Implementação

Este capítulo tem por objetivo descrever os elementos mais importantes ao processo de desenvolvimento do protótipo do CAMAW. O capítulo é dividido em quatro seções: ambiente de implementação, implementação do protótipo, protocolo de sincronização e protocolo de comunicação.

5.1 Ambiente de implementação

O algoritmo foi implementado utilizando-se a linguagem de programação Java, a partir da plataforma *Java Micro Edition* (Java ME). A Java ME é apropriada ao desenvolvimento de aplicações embarcadas voltadas para pequenos dispositivos de propósito específico, limitados em memória e processamento. Durante o processo de desenvolvimento do protótipo, o software gerado a partir do algoritmo foi testado e depurado utilizando-se a *Squawk virtual machine*, uma máquina virtual Java desenhada para dispositivos móveis e utilizada nos dispositivos sensores da plataforma Sun SPOT (*Sun Small Programmable Object Technology*). Esta máquina virtual tem a característica de permitir a portabilidade do código Java e também a execução isolada das aplicações, possibilitando a execução de uma ou mais aplicações sem que haja interferência de uma sobre a outra (SUN SPOT 2010).

A Figura 10 apresenta o diagrama com as camadas de software que executam em um nó Sun SPOT. No topo estão as aplicações dos usuários. Na base está a máquina virtual Squawk que é instalada diretamente sobre o *hardware*, ou seja, sem a intermediação de um sistema operacional. No meio existem várias bibliotecas, como a *espot_device* que fornece acesso aos dispositivos físicos do nó e rotinas básicas de I/O (entrada e saída), além de incluir o acesso ao protocolo de rádio da camada MAC. A biblioteca *multi-hop* provê protocolos de comunicação via rádio de alto nível e é responsável também por rotear pacotes para nós que não estão em contato direto (ao alcance do rádio). Em relação a este tipo de comunicação, a plataforma Sun SPOT disponibiliza por padrão o algoritmo *Link Quality Routing Protocol* – LQRP (PASCHOALINO E MADEIRA 2007) para realizar o roteamento *multi-hop*. Entretanto, é importante ressaltar que no presente trabalho foi utilizado o *Collection Tree Protocol* – CTP (GNAWALI et al., 2009) como protocolo de comunicação *multi-hop*, com o objetivo principalmente de minimizar o número de transmissões para esse tipo de comunicação, mas também por este protocolo estar disponível para plataforma Sun SPOT. Por fim, a bibiloteca *edemo_device* provê uma forma de acessar o *hardware* das placas de sensoriamento presentes no nó Sun SPOT, tais como acelerômetro, LEDs, chaves, I/O, entradas analógicas, etc.

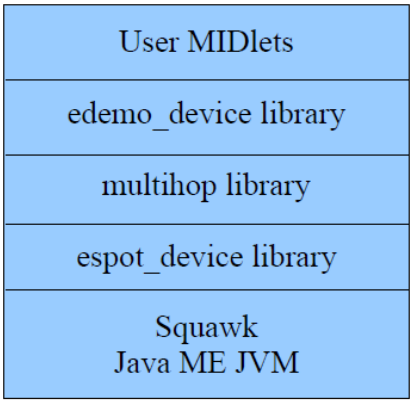


Figura 10. Diagrama da pilha de software em execução em um nó Sun SPOT

5.2 Implementação do protótipo

O CAMAW foi implementado em um único programa que compreende todos os componentes necessários (Figura 11) para que os nós Sun SPOT possam desempenhar tanto o papel de nó membro (CM), quanto o de nó líder (CH) em agrupamentos que compõem a organização da rede. O diagrama de componentes da implementação do CAMAW está ilustrado na Figura 11.

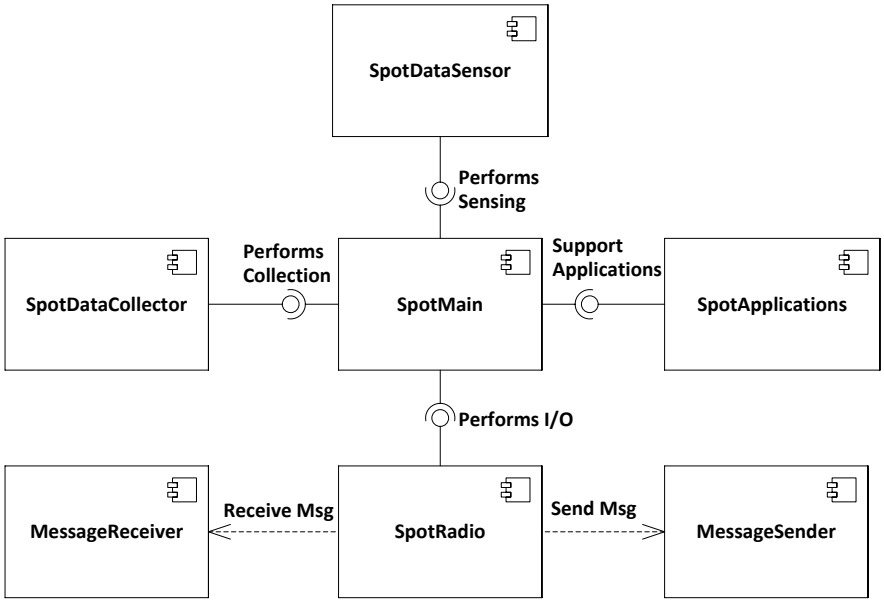


Figura 11. Diagrama de componentes para os nós da rede

A implementação do CAMAW é composta por um total de 7 componentes: **SpotMain, SpotDataSensor**, **SpotDataCollector**, **SpotApplications, SpotRadio**, **MessageReceiver** e **MessageSender**. O **SpotMain** é o componente principal do nó e atua como o gestor dos componentes **SpotDataCollector**, **SpotDataSensor, SpotApplications** e **SpotRadio** através das interfaces **PerformsSensing**, **PerformsCollection, SupportApplications** e **PerformsI/O** respectivamente. O componente **SpotDataSensor** é responsável, em nós com papel de CM,

por lidar com a configuração e operação da tarefa de sensoriamento a ser realizada pelos dispositivos sensores presentes no nó. Já o componente **SpotDataCollector** é o responsável, em nós com papel de CHs, por agregar os dados coletados pelo *cluster* que serão em seguida submetidos ao nó sorvedouro. O componente **SpotApplications** é responsável por manter as informações de cada aplicação ativa na rede e seus respectivos requisitos de monitoramento. O componente **SpotRadio** é responsável por coordenar a utilização do rádio para envio e recebimento de mensagens durante a comunicação com outros nós, utilizando para isso os componentes **MessageReceiver** e **MessageSender**. Estes componentes, por sua vez, são responsáveis por encapsular os detalhes da utilização do rádio durante o envio e recebimento de mensagens. Como, por exemplo, a definição da porta e tipo de protocolo de comunicação a serem utilizados. Em nós da plataforma Sun SPOT a utilização do rádio 802.15.4 é disponibilizada em duas opções de protocolo: RadioStream e RadioGram. O RadioStream é mais indicado para comunição confiável entre dois nós, já que é um protocolo que oferece conexão. O RadioGram realiza comutação de pacotes sem a garantia da ordem de chegada ou mesmo da entrega dos pacotes.

5.3 Protocolo de sincronização

A escolha do tipo do protocolo de sincronização mais adequado é, em geral, definida caso a caso, de acordo com a necessidade da aplicação quanto a precisão, acurácia, custo e complexidade (REYES et al., 2013). Entretanto, para manter uma base comum de comparação dos experimentos entre diferentes aplicações, foi escolhido o protocolo de sincronização *Flooding Time Synchronization Protocol* – FTSP (MAROTI et al., 2004). Este protocolo foi desenvolvido especialmente para se adaptar às aplicações que exigem precisão rigorosa na execução sobre plataformas de RSSFs limitadas em recursos. O protocolo FTSP utiliza baixa largura de banda e é robusto contra falhas de nós e links. Essa robustez é alcançada utilizando inundações periódicas de mensagens de sincronização, o que possibilita que o protocolo continue funcionando mesmo quando há uma alteração na topologia da RSSF. Outro importante motivo para a escolha do FTSP é que este foi viabilizado para a plataforma Sun SPOT a partir do trabalho de Reyes et al. (2013).

5.4 Protocolo de comunicação

Esta seção descreve o protocolo de comunicação do CAMAW. Este protocolo é apresentado na forma de um conjunto de mensagens que, no contexto de várias aplicações, permitem às RSSFs cumprir seus objetivos de monitoramento e ao mesmo tempo permitir o compartilhamento de seus recursos. Por meio das mensagens descritas nesta seção o

CAMAW habilita a RSSF a suportar a entrada e saída das aplicações da rede, além de permitir que os nós trabalhem colaborativamente em prol destas mesmas aplicações. A Tabela 2 descreve os tipos de mensagem utilizada no CAMAW, bem como seu identificador da mensagem, a origem e o destino da mensagem.

Tabela 2. Conjunto de mensagens do CAMAW

MENSAGEM	IDENTIFICADOR	ORIGEM	DESTINO
BS_NEW_APP	A	Sorvedouro	Todos
CAPABILITES_EXCHANGE	B	Todos	Todos
NODE_GRADING	C	Todos	Todos
NEW_COLLECTOR	D	CH	Todos
JOIN_CLUSTER	E	CM	CH
SENSED_DATA	F	CM	CH
AGGREGATED_DATA	G	CH	Sorvedouro
END_APPLICATION	H	Sorvedouro	CH
UPDATE_SENSORING	J	CH	CM

A mensagem BS_NEW_APP é submetida para a rede por meio do nó sorvedouro e é direcionada a todos os nós da rede visando informar a todos a chegada de uma nova aplicação. Conforme o exemplo da tabela 3, a mensagem contém, no mínimo, 10 campos com o seguinte formato: identificador da mensagem, quantidade de aplicações contidas na mensagem, identificador da aplicação, quantidade de unidades de sensoriamento requeridas pela aplicação, identificador da unidade de sensoriamento/taxa (frequência) de sensoriamento, área de interesse da aplicação definida por coordenadas (x_1,y_1) e (x_2,y_2). É importante mencionar que pelo fato do algoritmo ser desenhado para tratar múltiplas aplicações, optou-se pelo uso de caracteres separadores para delimitar os diferentes campos que compõem uma mensagem. Dessa forma, a quantidade de campos da mensagem pode variar de acordo com a quantidade de aplicações envolvidas na mensagem, evitando o envio de mensagens adicionais. Por isso, em cada mensagem existem campos que indicam a quantidade de aplicações envolvidas que servem como índices para que o programa possa iterar sobre a mensagem e extrair informações pertinentes a cada aplicação. No exemplo da mensagem BS_NEW_APP da tabela 3 verifica-se que o segundo campo, cujo valor é “3”, informa que no restante da mensagem existirão informações pertinentes a três diferentes aplicações e que, portanto, o programa deverá iterar pelo menos três vezes para recuperar as informações associadas a cada

aplicação. Tais informações consistem de unidades de sensoriamento, taxas de monitoramento e áreas de interesse requeridas para as aplicações 1,2 e 3.

Tabela 3. Descrição das mensagens do CAMAW

MENSAGEM	EXEMPLO
BS_NEW_APP	"A;3;1;5;13,23,33,43,53;14,20,45,48;2;5;13,23,33,43,53;14,20,45,48;"
CAPABILITIES_EXCHANGE	"B;2,2#2.0,5#1.0;27,47"
NODE_GRADING	"C;3,1,2,3;0.5,0.7,0.9"
NEW_COLLECTOR	"D;3,2,1,3"
JOIN_CLUSTER	"E;1;"
SENSED_DATA	"F;3;1;0.93;2;1;0.32"
AGGREGATED_DATA	"G;2;5;3,1,3,2;0.79;1;3,2,1,3;0.30"
END_APPLICATION	"H;3"
GO_TO_SLEEP	"K;2;1,2;"
UPDATE_SENSORING	"M;1;1;5;14,24,34,44,54;14,20;3;8;"

A mensagem CAPABILITIES_EXCHANGE serve para que os nós possam trocar entre si informações sobre as unidades de sensoriamento presentes em cada nó, além de sua localização no campo de sensoriamento. A mensagem NODE_GRADING é utilizada para que cada nó comunique a seus vizinhos a sua nota, elaborada com base na sua energia residual e quantidade de vizinhos vinculados a cada aplicação da rede. A mensagem NEW_COLLECTOR serve para que o nó se anuncie como CH aos seus vizinhos em relação a uma ou mais aplicações. A mensagem JOIN_CLUSTER serve para o nó se associar ao CH com melhor RSSI. A mensagem SENSED_DATA é enviada por nós CMs para os seus respectivos CHs e carregam dados resultantes do monitoramento realizado pelos CMs. A mensagem AGGREGATED_DATA serve para que nós CHs possam enviar dados agregados de monitoramento em direção ao nó sorvedouro. A mensagem END_APPLICATION é submetida para a rede pelo nó sorvedouro e serve para encerrar o monitoramento realizado para uma ou mais aplicações. A mensagem UPDATE_SENSORING serve para o nó CH informar o monitoramento requerido aos nós do agrupamento e também organizar a comunicação *intra-cluster* através de um esquema TDMA, informando a ordem e o período de tempo em que cada nó CM deve se comunicar.

6 Experimentos

Foram realizados experimentos em uma rede de sensores simulada, onde foi possível avaliar o impacto do algoritmo proposto sobre uma RSSF em termos de consumo de recursos de memória RAM e ROM, em termos comunicação, a partir do tráfego de pacotes de controle e de dados e em tempo de duração da rede em um ambiente com várias aplicações em execução simultânea.

O restante deste capítulo foi dividido em 7 itens, a saber: (i) ambiente dos experimentos, (i) métricas, (iii) cenários, (iv) experimentos para avaliar o consumo de recursos da rede,(v) experimentos para avaliar o *overhead* de pacotes trafegados na rede,(vi) experimentos para avaliar o tempo de vida da rede, (vii) análise comparativa entre o CAMAW e outros trabalhos.

6.1 Ambiente dos experimentos

Esta seção de ambiente de experimentos foi dividida em ambiente de simulação e ambiente com sensores reais. Em ambos foi utilizado o emulador Solarium que permite que sejam realizadas simulações com um grande número de nós virtuais em conjunto com nós sensores reais. Durante as simulações foram utilizados 3 nós sensores reais da plataforma Sun SPOT.

O Sun SPOT é um dispositivo sensor desenvolvido pela Sun Microsystems em 2007 composto por um processador ARM (*Advanced Risc Machine*) de 400 MHz, com 1MB de memória RAM e 8MB de memória flash. Também é equipado com um rádio IEEE 802.15.4 para comunicação sem fio que opera na frequência de 2.4 GHz que possui um alcance máximo de 80m. Possui também uma bateria recarregável de 3,7 V, 770 mAh de Íon-Lítio para fornecer um suprimento de energia que pode durar mais de 909 dias quando no modo máximo de economia de energia (REYES et al., 2013). Cada um destes dispositivos SUN SPOT contém uma placa que inclui sensores de temperatura, luminosidade e movimento (acelerômetro) e além disso, também possuem pinos de entrada e saída de propósito geral para que possam ser incluídos sensores adicionais ou outro *hardware*. Uma característica que torna o SUN SPOT um candidato ideal para soluções embarcadas é a capacidade de executar em cada spot uma máquina virtual Java ME completa. Java, por sua vez, é uma linguagem de alto nível orientada a objeto que oferece interoperabilidade de software entre diferentes plataformas e sistemas operacionais. O Java, bem como suas IDEs NetBeans e Eclipse, estão disponíveis gratuitamente em diferentes sistemas operacionais, tais como Mac OS, Windows e Linux. Por conta de tudo isso, a linguagem Java é particularmente adequada para

prototipagem rápida de aplicações de RSSF no ambiente SDK (*Software Development Kit*) do Sun SPOT.

O Solarium, aplicativo que faz parte da plataforma SUN SPOT, provê um emulador de SPOT que é útil para realizar testes com software desenvolvido para o SPOT e/ou para realizar experimentos com um grande número de nós, em situações nas quais o *hardware* real não está disponível. O Solarium é um emulador confiável, pois a partir do mesmo código fonte, é possível instalar e conectar em um mesmo ambiente de simulação, tanto nós virtuais quanto nós reais, conferindo credibilidade aos experimentos. Outra importante funcionalidade do Solarium é a geração de *logs* de execução que permitem a extração de dados relevantes sobre a execução e o desempenho do software embarcado no contexto de cada nó sensor da rede.

O desenvolvimento e a implementação do CAMAW foram realizados a partir da plataforma Sun SPOT de sensores. Apesar de o CAMAW ser desenhado para lidar com dados oriundos de qualquer tipo de unidade de sensoriamento, não foram utilizadas, durante os experimentos, as unidades físicas presentes nos sensores Sun SPOT (temperatura, luminosidade e movimento), para não haver diferenças entre nós reais e virtuais na tarefa de coleta dos dados. Por isso, foi adotado um gerador randômico para simular os dados obtidos nas diferentes unidades de sensoriamento. É importante mencionar que as tarefas de fusão de dados e roteamento *multi-hop* em direção ao nó sorvedouro não foram definidas como sendo da responsabilidade do CAMAW.

Figura 12. Dispositivo SUNSPOT

6.1.1 Ambiente de simulação

As simulações foram realizadas em uma máquina Intel Core i7 de 2.9 GHz CPU e 8 GB RAM. Foi utilizado o Emulador Solarium versão 1.2.0 para testes e validação do algoritmo em um ambiente onde os nós sensores da plataforma Sun SPOT são emulados. Através do Solarium foi possível estimar também o consumo de energia dos nós associado às atividades de comunicação e sensoriamento.

6.1.1.1 Ambiente com nós sensores reais

O emulador Solarium permite a portabilidade de software, pois a partir do mesmo código fonte é possível, sem nenhum esforço adicional de programação, realizar experimentos tanto com nós virtuais quanto nós reais. Portanto, o Solarium além de ser uma ferramenta útil nos testes de software por emular grupos de nós virtuais, também é capaz suportar a operação combinada com nós reais em uma mesma rede, como pode ser observado no diagrama da Figura 13. Para tanto basta configurar o nó sorvedouro modificando o atributo *"basestation.shared = True"* no arquivo de propriedades *.sunspot.properties*. Os dados das simulações que interessam aos experimentos foram obtidos através da capacidade do emulador mencionada anteriormente de gerar *logs* de execução.

O ambiente com nós sensores reais foi montado em um ambiente controlado, dentro do Laboratório de Redes sem Fio (LabNet) do Programa de Pós-Graduação em Informática da UFRJ (PPGI). Foram utilizados 3 nós Sun Spot reais mais um o nó sorvedouro acoplado via cabo USB a um computador desktop.

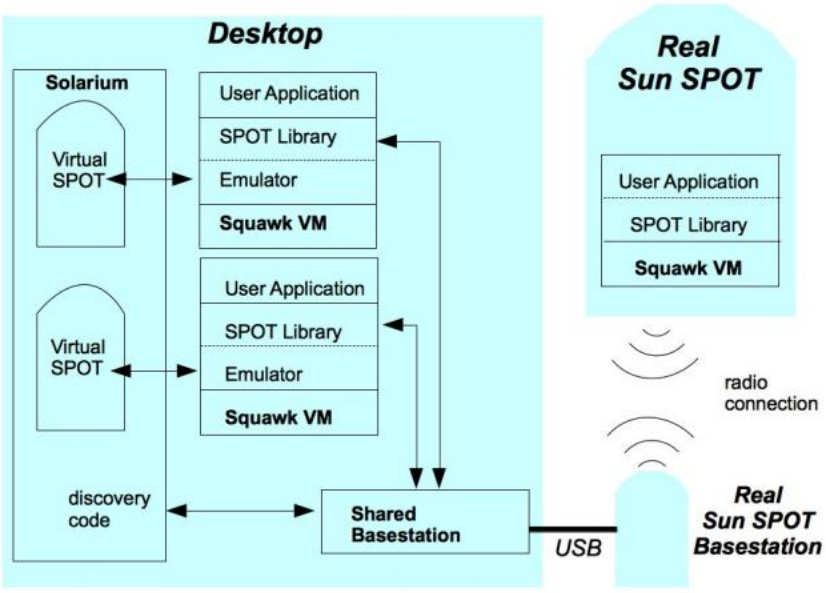


Figura 13. Diagrama de blocos com a arquitetura do emulador Solarium, contendo nós reais e virtuais.

Os experimentos com nós reais tem a finalidade de verificar a aplicabilidade real do algoritmo, na medida em que estes nós, juntamente com os nós virtuais, também são capazes de executar todas as fases do algoritmo, contribuindo assim para a organização da rede. Isso proporciona confiabilidade aos experimentos para que possa ser realizada a implantação do algoritmo em redes reais de nós sensores.

6.1.2 Modelo de energia

O estado de um nó sensor pode ser descrito pela combinação de estados dos módulos de comunicação, sensoriamento e processamento. Foi definido que cada módulo funcional possui apenas dois estados de operação: ativo e inativo. Cada módulo só pode estar em um dos dois estados e o seu estado não afeta o estado dos outros dois módulos. Os módulos de sensoriamento e de comunicação são os módulos mais relevantes em relação ao consumo energético. O consumo energético é então definido como uma função do estado do nó e o tempo que este permanece naquele estado. Quando o estado do nó é determinado, este consome uma taxa fixa de energia durante aquele período de tempo. Foi considerado que, para os experimentos, o gasto energético dos módulos quando em estado inativo é irrelevante sendo, portanto, considerado 0 para qualquer período de tempo. Quando o nó está em estado ativo o consumo de energia durante um período t é calculado pela seguinte fórmula $E(T) = E_c + E_s$, o, onde E_c representa o consumo de energia do módulo de comunicação, e E_s representa o consumo de energia do módulo de sensoriamento.

Para avaliar o consumo da comunicação entre os nós da rede é preciso ter em mente que exceto o nó sorvedouro e os nós CHs todos os outros nós (CMs ou ainda sem papel definido) só trocam mensagens através de comunicação direta (ao alcance do rádio). O consumo de energia de transmitir 1-bit pela rede por uma distância *d* é definido como $E_{tx}(l, d)$:

$$E_{tx}(l, d) = E_{elec} \times l + \varepsilon_{amp} \times l \times d^2 \qquad \textbf{(5)}$$

Onde *Eelec* e *εamp* são parâmetros relacionados ao *hardware* (MAHAJAN & MALHOTRA, 2011). Foi observado durante as simulações que o gasto energético do receptor durante o processo de troca de mensagens é irrelevante e, portanto, foi desconsiderado. O consumo de energia associado à transmissão de 1-bit de dados da origem até o destino em comunicações multi-salto em direção ao nó sorvedouro é definido como $E_{tx}(l, src, des)$:

$$E_{tx}(l, src, des) = \sum_{i=1}^{k} E_{tx}(l, d) \qquad \textbf{(6)}$$

Onde k é a quantidade mínima de saltos que o dado percorre da origem até o destino e $E_{tx}(l,d)$ permanece com o mesmo significado quando mostrado na equação (13). O gasto de energia referente ao módulo de sensoriamento é calculado pela equação linear:

$$E_{s_i} = ER_{s_i} \times t_{s_i} \qquad (6)$$

Onde $\mathrm{ER_{s_i}}$ representa o consumo de energia do módulo *i* durante a unidade de tempo $\mathrm{t_{s_i}}$ que representa o período de tempo da execução de *i.*

6.2 Métricas

Para avaliar o CAMAW quanto ao consumo de recursos foram definidas as métricas: **porcentagem de bytes ocupados em memória RAM (PR)** e **porcentagem de bytes ocupados em memória Flash (PF)**. Para avaliá-lo quanto ao *overhead* de mensagens na rede foram definidas as métricas: **pacotes de controle trafegados (PCT)** e **pacotes de dados trafegados (PDT)**. Por fim, para avaliá-lo quanto à a duração da RSSF foi definida a métrica **tempo de vida da rede (TVR)**.

O **Tempo de vida da rede (TVR)** é uma métrica importante, pois nós sensores possuem recursos limitados de baterias que são normalmente não recarregáveis. Portanto quanto maior for o tempo de vida dos sensores, maior será o tempo no qual a rede permanecerá operando e atendendo seus objetivos de monitoramento. Existem diferentes maneiras de avaliar o tempo de vida de uma rede em algoritmos de clusterização (primeiro nó a morrer - FND, metade dos nós vivos - HNA, último nó a morrer - LND, etc.). Porém para este trabalho foi adotada a FND medida em rodadas, ou seja, a quantidade de rodadas até o primeiro nó na RSSF esgotar a sua energia ao ponto de ser incapaz de realizar as operações de sensoriamento e/ou comunicação.

O **número de pacotes de controle transmitidos (PCT)** é definido pelo somatório de pacotes de controle enviados por todos os nós da rede. Um pacote é dito de controle quando sua carga útil é composta por informação de controle relacionada à organização rede. Segundo a seção 5.4 as mensagens com este tipo de pacote são as que possuem os identificadores “A”,”B”,”C”,”D”,”E”,”H”,”I”,”J”,”K”,”L”,”M”. A PCT é aferida ao final de cada experimento somando-se o montante de pacotes de controle transmitidos por cada nó rede.

O **número de pacotes de dados transmitidos (PDT)** é definido pelo somatório de mensagens de dados enviadas por todos os nós da rede. Um pacote é dito de dados quando sua carga útil é composta de dados de monitoramento produzidos pela operação da rede, ou seja, composta por informação obtida pelos dispositivos de sensoriamento presentes em cada nó sensor. Segundo a seção 5.4 as mensagens com este tipo de pacote são as que possuem os identificadores “F” e “G”. Da mesma forma que a PCT, a PDT é aferida ao final de cada experimento com base no total de pacotes de dados transmitidos por cada nó da rede.

A **Porcentagem de memória RAM ocupada (PR)** é definida pela fração da memória RAM que em média fica ocupada com o algoritmo em execução. A TR é medida somente em nós do tipo CH pelo fato de estes serem os mais exigidos em termos do consumo de recursos de memória decorrente da sua responsabilidade em coletar os dados do agrupamento. Para se chegar a este resultado primeiro é obtida, para cada nó CH, a razão entre a memória RAM ocupada com o *software* do algoritmo em funcionamento e a quantidade total de memória RAM disponível para execução de aplicações em nós da plataforma Sun SPOT. Em seguida somam-se esses valores de cada nó da rede e divide-se pela quantidade total de nós. Isso significa que a TR descreve a taxa média de memória RAM ocupada, em termos percentuais, pelos CHs da rede.

A **Porcentagem de memória Flash ocupada (PF)** é definida pela fração, em termos percentuais, de memória *Flash* ocupada pelo *software* compilado do algoritmo em relação à quantidade total de memória *Flash* disponível para o armazenamento de aplicações em nós da plataforma Sun SPOT.

6.3 Cenários

Esta seção define um cenário padrão para a realização dos experimentos. A RSSF é composta por um cenário base composto por 100 nós fixos em posições escolhidas aleatoriamente em um campo de sensoriamento de 100 m x 100 m. Cada nó contém 2 unidades de sensoriamento diferentes escolhidas também aleatoriamente entre 5 possíveis (1-temperatura, 2-luminosidade, 3-umidade, 4-pressão e 5-presença). A localização geográfica de cada nó e suas respectivas unidades de sensoriamento estão descritas no Anexo 1 deste trabalho. Estas informações descrevem o cenário base utilizado em todos os experimentos. A Figura 14 exibe graficamente o posicionamento geográfico dos nós com base nas informações do Anexo 1. Durante os experimentos as baterias de cada um destes nós são carregadas com uma energia inicial de 0.5 *Joules*. O único nó sorvedouro foi posicionado nas coordenadas (100,0). Definiu-se que cada aplicação requer 2 unidades de sensoriamento distintas, também

escolhidas de forma aleatória e com taxas de monitoramento que variam de 1 a 5 segundos, conforme o procedimento descrito em Raghunathan et al. (2002). No CAMAW cada aplicação define uma região específica do campo de sensoriamento a ser monitorada. Esta região é definida através de coordenadas como em um plano cartesiano bidimensional (x, y). Mas para os experimentos realizados neste trabalho, todas as aplicações tem interesse na área total do campo de sensoriamento. Cada experimento tem início com a chegada da primeira aplicação na rede através do nó sorvedouro. O término de cada experimento é caracterizado pelo evento do primeiro nó a morrer na rede em função do esgotamento de sua bateria. Isso ocorre quando a energia residual do nó não é suficiente para a realização das operações básicas de sensoriamento e/ou comunicação. Como o nó sorvedouro pode difundir mensagens para a rede a qualquer momento, o rádio de cada nó da rede fica permanentemente ligado para receber mensagens a qualquer instante.

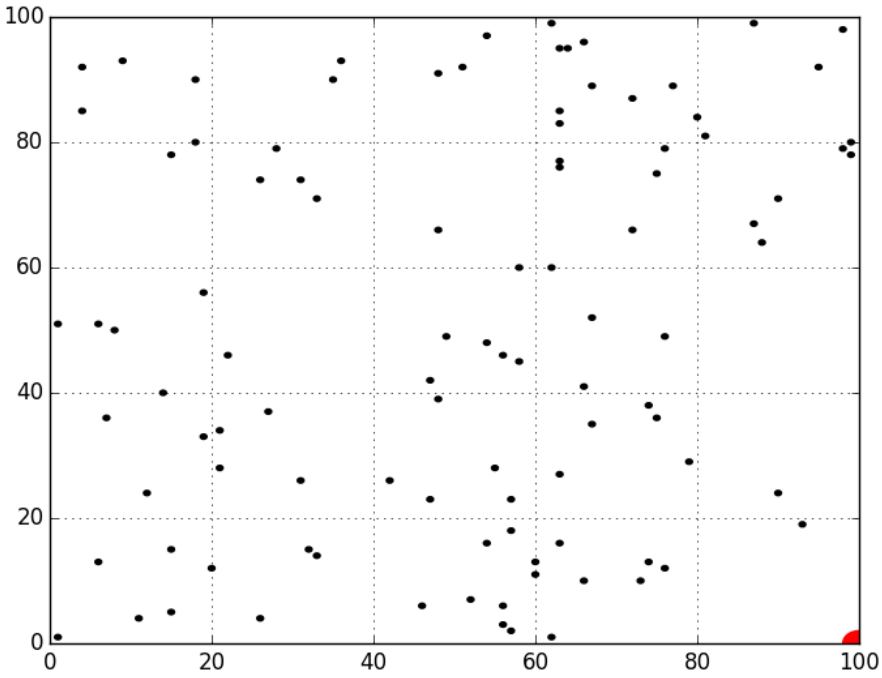


Figura 14. Representação geográfica da distribuição dos nós sensores no campo de monitoramento

Alguns parâmetros comuns em algoritmos de clusterização foram fixados neste trabalho por não estarem diretamente relacionados aos objetivos desta proposta vinculados ao tratamento de múltiplas aplicações: *node degree*, que é a quantidade de nós em um *cluster* e neste trabalho é definido como o total de nós aptos dentro do perímetro do *cluster*; e o raio do *cluster*, definido como o alcance de rádio do CH, que foi fixado tanto em nós CHs quanto em nós CMs em 30 metros. As simulações foram conduzidas dentro de um ambiente controlado de laboratório e livre de ruído, portanto não foram consideradas perdas de pacotes durante os

experimentos. Por fim, os resultados apresentados representam a média de 30 repetições para um intervalo de confiança de 95%.

6.4 Experimentos para avaliar o consumo de recursos da rede

Para avaliar o CAMAW em termos de consumo de recursos da rede foram utilizadas as métricas **Percentual de memória RAM ocupada** (**PR)** e **Percentual de memória Flash ocupada (PF)**. Para se quantificar a TR foi realizado um conjunto de experimentos tomando como base o cenário descrito na seção 6.3. Neste item foram realizados experimentos com números diferentes de aplicações simultâneas na rede (2, 4, 6, 8 e 10). O evento de entrada das aplicações na rede caracteriza o início do experimento e o evento da morte do primeiro nó da rede indica o final do experimento. As aplicações permanecem em execução na rede enquanto durar o experimento. A Figura 15 exibe os resultados da PR obtidos para o conjunto de experimentos realizados. As informações para o cálculo da PR são obtidas a partir dos métodos *totalMemory()* e *freeMemory()* presentes na classe *Structure* que está disponível na API java com.sun.cldc.jna.

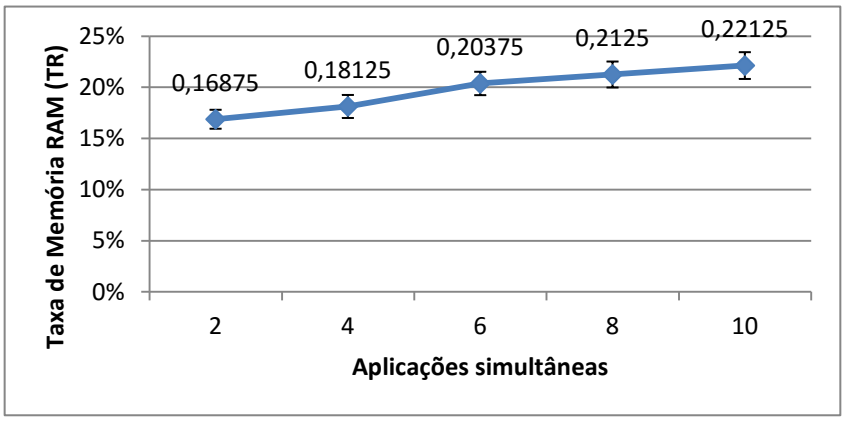


Figura 15. Consumo de recursos de memória RAM em nós da rede

O gráfico indica uma correlação entre o crescimento do número de aplicações simultâneas na rede e a PR, pelo fato de que cada nova aplicação representa naturalmente mais um impacto no consumo dos recursos da rede. Para a simulação com 2 aplicações o CAMAW apresentou uma PR média de 16,88% com um desvio padrão de 9,25% enquanto que para a com 10 aplicações o CAMAW apresentou uma PR média de 23,13% com um desvio padrão de 13,10%. Embora esteja clara a correlação entre o número de aplicações e a PR, o seu crescimento não é linear. A Figura 15 demonstra que o impacto adicional da PR é cada vez menor conforme aumenta-se o número de aplicações. Isso se deve a dois motivos: primeiro, pelo fato do CAMAW possuir uma visão única da rede, por causa da abordagem de instância simples que contém todas as aplicações, o impacto em memória de uma nova

aplicação se resume a poucos atributos da instância da classe *AppIDRequirements*; segundo, porque o CAMAW aproveita as redundâncias nos dados requisitados pelas aplicações para consumir menos memória como, por exemplo, no caso de 3 aplicações interessadas no monitoramento da temperatura. Não há, nesse caso, a necessidade de 3 pilhas de dados de temperatura no CH, uma para cada aplicação. Portanto, apenas uma única pilha é criada e seus dados são compartilhados entre as três aplicações. Os resultados dos experimentos demonstram que o algoritmo não onera excessivamente, em termos percentuais, a memória RAM do CH. Além disso, até mesmo em simulações com 10 aplicações simultâneas ainda há em média 76,87 % de memória livre para comportar a chegada de novas aplicações.

Já para a obtenção da PF basta realizar uma leitura diretamente da memória secundária, no caso deste sensor da memória *Flash*, para se verificar o espaço ocupado pelo código objeto correspondente ao algoritmo. Pode-se observar que o CAMAW ocupa apenas 64 Kb da memória Flash do nó sensor. Como os nós da plataforma Sun SPOT versão REV8 possuem um total de 8M bytes de memória Flash, sendo que desse total 800 Kb são reservados pelo sistema para armazenar a área de boot, tabela de alocação de arquivos, máquina virtual, arquivos de sistema, arquivos de propriedades, entre outros; apenas o restante da memória (7200 Kb) está disponível para o armazenamento de aplicações e seus respectivos arquivos. Portanto o valor da PF é igual a 0,89%.

Os valores da PR e da PF indicam que, considerando-se todos os experimentos realizados onde foram variados o número de aplicações em execução, o CAMAW se mostrou uma solução escalável. Isso ocorre porque o CAMAW produz baixo impacto sobre os recursos da rede, na medida em que onera pouco, em termos percentuais, tanto a memória RAM quanto a memória Flash do nó sensor.

6.5 Experimentos para avaliar o overhead de pacotes trafegados na rede

Nesta seção foram utilizados dados coletados durante o conjunto de experimentos realizados na seção 6.4 para se avaliar também o overhead de pacotes trafegados pelo CAMAW em termos de **pacotes de controle trafegados (PCT)** e **pacotes de dados trafegados (PDT)**. Os pacotes foram contabilizados em cada nó da rede do início até o final de cada experimento. A Figura 16 exibe o somatório de PCT e de PDT nos experimentos com 2, 4, 6, 8 e 10 aplicações.

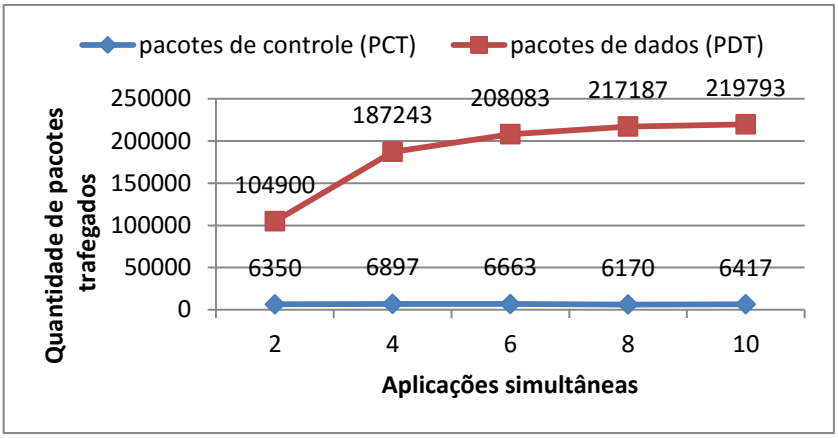


Figura 16. Pacotes trafegados pelo CAMAW

A Figura 16 exibe os valores de PCT que indicam uma pequena quantidade de mensagens em relação a PDT e que se mantém praticamente estável conforme se varia o número de aplicações. Isso se deve ao fato do CAMAW ser um algoritmo de tempo de convergência constante (ABBASI e YOUNIS, 2007), ou seja que organiza a rede após um número fixo de iterações. Isso significa que a rede estará organizada em agrupamentos de nós após uma sequência pré-definida de pacotes de controle enviados por cada nó apto. As pequenas variações da PCT observadas podem ser explicadas pela variação do número de CHs e de CMs entre uma rodada e outra, pelo fato de que nós CHs e CMs trafegam quantidades diferentes de pacotes de controle, conforme detalhado no capítulo 4.

Quanto aos pacotes de dados PDT, o comportamento observado na Figura 16 reflete a capacidade do CAMAW em identificar e tirar proveito das comunalidades entre diferentes aplicações. A forma como foram modeladas as aplicações, restritas a 2 unidades de monitoramento cada qual praticando uma taxa de monitoramento dentre 5 possíveis: 1, 2 ,3 ,4 ou 5 segundos; favorece a ocorrência de redundâncias entre aplicações conforme explicado na subseção 4.4.4. Sendo assim, nos experimentos com 2 aplicações, o tráfego de mensagens é menor, pois podem ocorrer no máximo 4 taxas distintas de sensoriamento entre as 5 possíveis. Isso faz com que a média de mensagens trafegadas durante as simulações seja inferior à taxa máxima possível (1msg/s). Entretanto, em simulações com mais aplicações, há uma probabilidade maior das 5 taxas possíveis serem requisitadas, fazendo com que a taxa média se aproxime da taxa máxima possível. Então, se por um lado, aumentar o número de aplicações significa aumentar a taxa média de mensagens trafegadas, por outro lado também significa aumentar a chance de ocorrerem requisitos comuns de sensoriamento entre aplicações, criando oportunidades para a redução do número de mensagens trafegadas. Isso explica o formato da curva que corresponde à PDT na Figura 16. A redução da quantidade de

mensagens é proporcional ao número de aplicações concorrentes com interesse na mesmas unidades de sensoriamento.

A Figura 16 indica que conforme se aumenta o número de aplicações simultâneas durante os experimentos, a PDT tende a estabilizar. O comportamento demonstrado na Figura 16 sugere que o CAMAW é uma solução escalável, já que o tráfego de mensagens na rede (PDT e PCT) tende a se estabilizar com a entrada na rede de um número maior de aplicações simultâneas.

6.6 Experimentos para avaliar o tempo de vida da rede

Nesta seção foram utilizados dados coletados durante o conjunto de experimentos do item 6.4 com o objetivo de avaliar o desempenho do CAMAW quanto a duração da RSSF, ou seja, o período total de operação da rede segundo a métrica de **tempo de vida da rede (TVR)**. Os resultados desta seção são particularmente importantes, pois o tempo de vida da rede constitui um dos itens mais avaliados nos *softwares* voltados para as RSSF. Neste trabalho a TVR indica por quanto tempo a rede é capaz de permanecer em operação em um ambiente com múltiplas aplicações simultâneas. Estas informações serão utilizadas também mais adiante como base de comparação com outros algoritmos da literatura.

Com base nos dados da Figura 17 é possível afirmar, conforme o número de aplicações simultâneas aumenta, os valores de TVR diminuem. Entretanto a forma da curva da TVR indica que há uma crescente economia de energia conforme se incrementa a quantidade de aplicações simultâneas. De acordo com o comportamento do CAMAW também observado nas seções 6.4 e 6.5, percebe-se que o acréscimo na quantidade de aplicações simultâneas resulta em uma maior probabilidade de requisitos comuns de monitoramento entre aplicações. O CAMAW explora estas oportunidades evitando, no nível dos nós da rede, a repetição desnecessária de atividades de monitoramento sempre que um dado coletado em um determinado instante interessar a mais de uma aplicação. Além disso, o CAMAW também transmite este dado uma única vez, de forma a endereçar todas as aplicações interessadas. Estas duas funcionalidades presentes no CAMAW contribuem para um bom desempenho em termos de duração da rede que pode ser observado no gráfico da Figura 17, considerando-se que na abordagem ingênua, com estas funcionalidades ausentes, seria esperado um decaimento linear e mais acentuado na duração da rede.

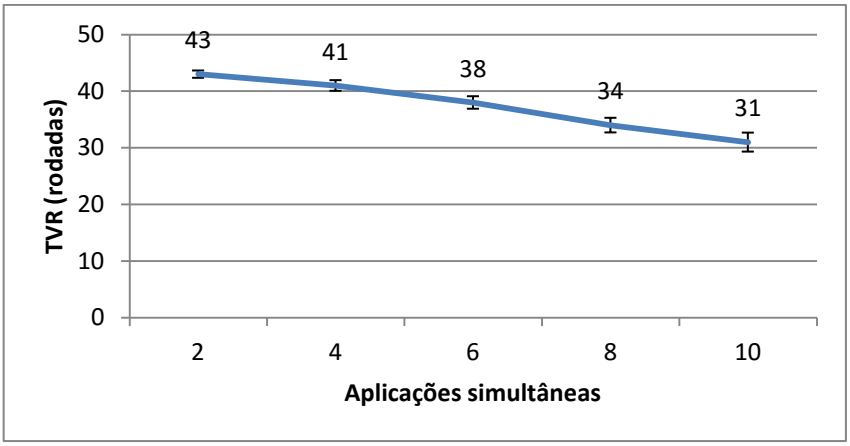


Figura 17. TVR em rodadas do CAMAW com diferentes números de aplicações.

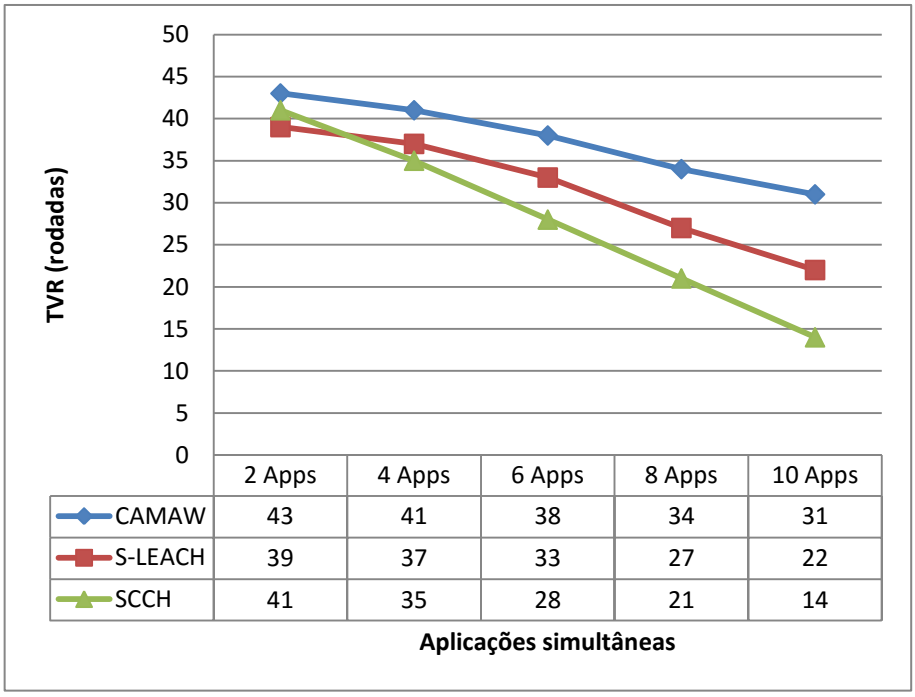
Os resultados demonstram que nos experimentos com 2 aplicações houve um TVR de 43 rodadas para uma bateria de 0,5 Joule. Já para o cenário mais exigente com 10 aplicações o TVR do CAMAW é de 31 rodadas utilizando os mesmos 0,5 Joule de bateria. Considerando que a bateria de um sensor Sun SPOT possui 720 mah. Através da fórmula da lei de Ohm $P = V * i$ verifica-se que a potência é um produto entre a tensão elétrica e a corrente elétrica. A partir desta fórmula e com a componente corrente expressa em ampere-hora é possível se obter a quantidade de watt-hora que representa a energia residual da bateria do nó sensor. Para expressá-la em *Joules*, basta se utilizar a equivalência 1Wh = 3600 *Joules*. Dessa forma, a carga total da bateria de 720 mah do sensor Sun SPOT possui uma reserva energética de 9590 *Joules*. Considerando os experimentos realizados, é possível afirmar que o CAMAW é capaz de durar 824.740 rodadas para o cenário com 2 aplicações e 594.580 rodadas para o cenário com 10 aplicações.

Com base nos valores obtidos para a métrica TVR para o conjunto de experimentos realizados é possível afirmar que o CAMAW é um algoritmo capaz de estender o TVR da rede de forma proporcional à economia obtida com a eliminação tarefas redundantes de monitoramento entre aplicações.

6.7 Análise comparativa entre o CAMAW e outros trabalhos

Foram escolhidos dois trabalhos da literatura para comparar seus resultados com os resultados obtidos pelo CAMAW. São eles o S-LEACH (CALDAS et al., 2015) e o SCCH (IZADI et al.,2015). Estes experimentos tem o objetivo de evidenciar a eficiência energética do CAMAW em relação ao S-LEACH e ao SCCH em termos de duração da rede que é um dos aspectos mais importantes e portanto mais avaliados no âmbito das RSSFs (YICK et al., 2008; MITRA et al., 2012). A partir deste conjunto de experimentos foram extraídos os

valores para a métrica TVR, escolhida para avaliar em termos quantitativos o desempenho de cada algoritmo. Foram realizados experimentos para os algoritmos com diferentes números de aplicações simultâneas na rede (2, 4, 6, 8 e 10). Ou seja, o número de aplicações definido no início de cada experimento é mantido inalterado até o seu final. A área de interesse de cada aplicação foi definida como a área total do campo de sensoriamento.



	2 Apps	4 Apps	6 Apps	8 Apps	10 Apps
CAMAW	43	41	38	34	31
S-LEACH	39	37	33	27	22
SCCH	41	35	28	21	14

Figura 18. Comparativo do TVR dos algoritmos com diferentes números de aplicações.

Os resultados (Figura 18) indicam que conforme o número de aplicações em execução simultaneamente na rede aumenta, os valores de TVR diminuem em todos os algoritmos, porém com uma diminuição muito menor no caso do CAMAW. O resultado da TVR para os experimentos com apenas 2 aplicações, foi de 43 rounds para uma bateria de 0,5 Joule. Este resultado é 10,26% melhor que o S-LEACH e 4,88% melhor que o SCCH. Já para os experimentos com mais aplicações (10) o CAMAW apresentou um TVR de 31 rounds com os mesmos 0,5 Joule de bateria. Este resultado é 40,91% melhor do que o apresentado pelo S-LEACH e 221,43% melhor do que o apresentado pelo SCCH.

6.7.1 Considerações finais sobre as comparações com outros trabalhos

Os resultados auferidos demonstram um desempenho superior do CAMAW em termos de duração da rede quando comparado ao S-LEACH e ao SCCH. Este desempenho decorre das características de seus *clusters* que promovem o balanceamento de carga da rede a partir de uma distribuição geográfica uniforme. Estes agrupamentos contêm apenas nós necessários

ao monitoramento requisitado o que significa que os demais nós da rede permanecem em baixo ciclo de trabalho para economizar energia. Além disso, cada agrupamento é capaz de eliminar redundâncias entre aplicações acerca das tarefas de comunicação e monitoramento, conforme explicado na subseção 4.4.4. Estas características dos agrupamentos do CAMAW permitem uma redução do consumo energético, fazendo com que a rede dure mais tempo. A Figura 19 exibe graficamente a rede organizada em agrupamentos no CAMAW durante uma rodada de monitoramento. Os nós de cor cinza não participam dos agrupamentos pois não foram considerados não aptos para as aplicações em execução e por isso permanecem em um baixo ciclo de trabalho, economizando energia até a chegada de novas aplicações.

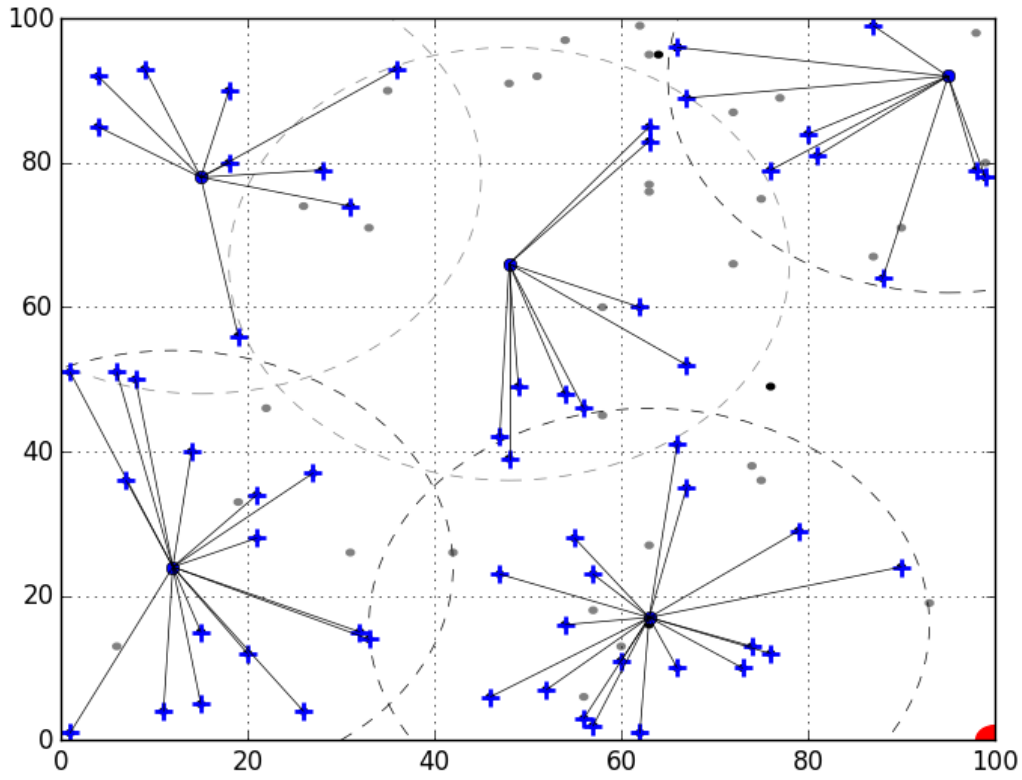


Figura 19. Distribuição geográfica dos agrupamentos no CAMAW durante uma rodada

O S-LEACH, assim como o CAMAW, também procura identificar e eliminar redundâncias entre o monitoramento das aplicações. Entretanto no S-LEACH a distribuição geográfica dos *clusters* pela rede é aleatória pelo fato de que a escolha se o nó será ou não CH é feita com base no resultado de uma função probabilidade. Isso pode levar a situações de sobreposição de *clusters*, áreas de sombra ou sem cobertura, além de *clusters* sobrecarregados. Essas situações resultam em um consumo não uniforme dos recursos da rede, o que leva a uma diminuição do seu tempo de vida. A Figura 21 exibe a rede organizada em agrupamentos no S-LEACH durante uma rodada de monitoramento.

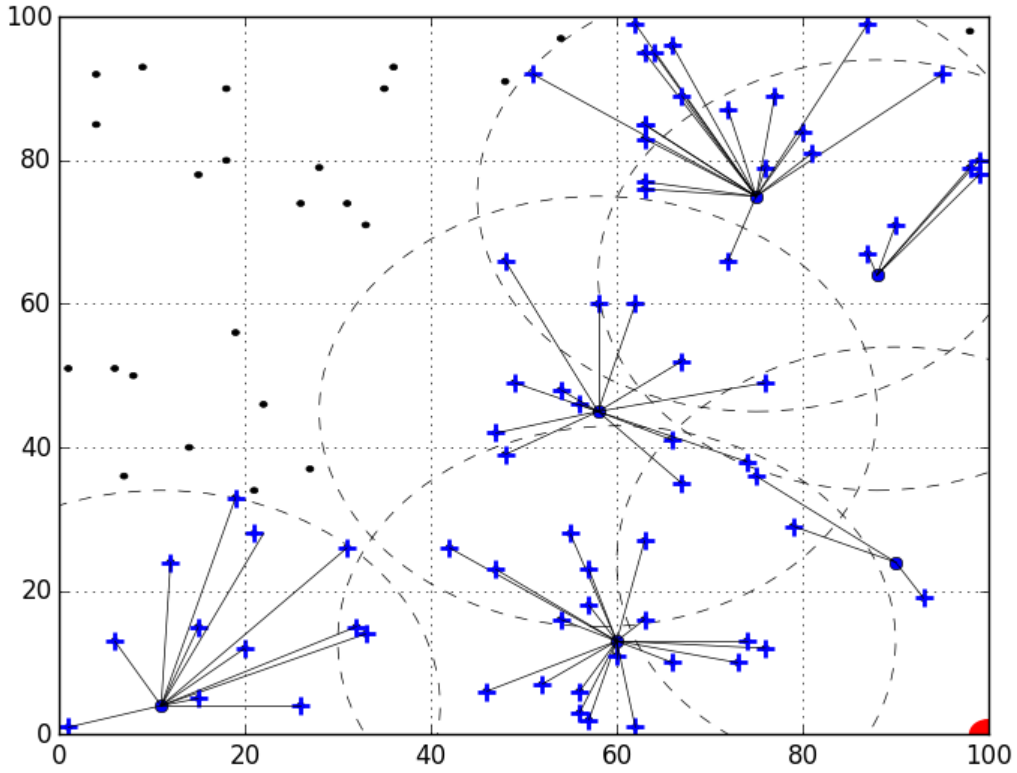


Figura 20. Distribuição geográfica dos agrupamentos no S-LEACH durante uma rodada

Já o SCCH promove a estabilidade da rede e uma distribuição geográfica dos *clusters* que considera os conceitos de centralidade do nó e distância local no processo de escolha do *cluster-head*. O objetivo é obter uma distribuição ótima dos clusters pela rede do ponto de vista do consumo energético. Entretanto o tratamento das aplicações é realizado isoladamente, o que impossibilita o uso de estratégias que tirem proveito de eventuais redundâncias no monitoramento entre aplicações. A Figura 22 exibe a rede organizada em agrupamentos no SCCH durante uma rodada de monitoramento.

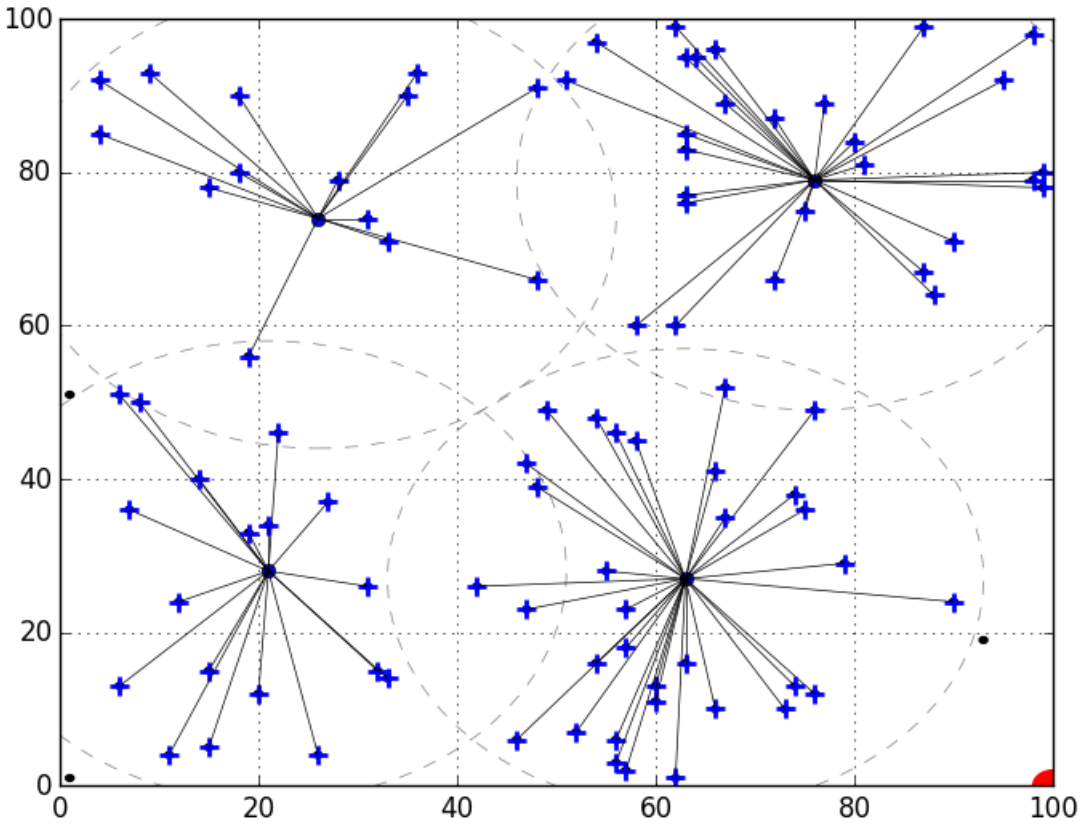


Figura 21. Distribuição geográfica dos agrupamentos no SCCH durante uma rodada

No SCCH e no S-LEACH as aplicações não interferem no procedimento de formação dos *clusters* que por isso, abrange toda a rede. Tal abordagem pode levar a um desperdício de energia, sempre que não for necessário o monitoramento de determinadas regiões do campo de sensoriamento. O CAMAW, por outro lado, só cria um agrupamento de nós conforme os requisitos geográficos definidos pelas aplicações e a partir de nós asseguradamente com a capacidade de monitoramento requerida pelas aplicações, em outras palavras, se por acaso o CH escolhido não apresentar em seu agrupamento todas as unidades requeridas, o agrupamento não será criado e será escolhido outro CH cujo grupo de sensores atenda aos requisitos da aplicação.

A partir dos resultados dos experimentos realizados nesta seção e com base nas características de cada algoritmo descritas acima é possível afirmar que em relação ao S-LEACH e ao SCCH, o CAMAW foi o algoritmo mais eficiente em relação a duração da rede em todos os experimentos com múltiplas aplicações realizados.

7 Conclusão

O presente trabalho propôs um novo algoritmo de clusterização distribuído denominado CAMAW que através do compartilhamento de dados é capaz de economizar recursos computacionais e de comunicação com o objetivo de conciliar os interesses de monitoramento de diversas aplicações concorrentes no âmbito das RSSFs.

É possível enumerar três grandes contribuições deste trabalho. Como primeira contribuição, este trabalho é o primeiro algoritmo de clusterização voltado para múltiplas aplicações a definir o parâmetro "região geográfica" como um dos requisitos da aplicação no processo de organização da rede. Ao contrário das abordagens tradicionais que em geral clusterizam toda a rede, o fato de restringir a área da RSSF a ser monitorada pode produzir uma grande economia de energia, pelo fato de que os nós que não estão envolvidos no monitoramento podem permanecer num estado de baixo consumo de energia.

Como segunda contribuição o CAMAW procura, durante a formação e operação dos seus agrupamentos, identificar e eliminar redundâncias nos requisitos entre as aplicações, reduzindo o gasto energético da RSSF nas operações de sensoriamento e de comunicação.

Como terceira contribuição o CAMAW é um algoritmo que habilita a RSSF no suporte a diferentes modelos de entrega de dados, o que torna a rede apta a receber uma maior variedade de aplicações em termos de necessidades de monitoramento. Estas aplicações podem ser classificadas, segundo a classificação proposta em Borges et al.(2014), em 4 modelos distintos de entrega de dados: *query-driven*, *continuous-based*, *time-driven* e *data-driven*.

Com relação às contribuições referentes a produção científica, partes desta dissertação foram publicados em um congresso internacional através do seguinte artigo:

(COSTA et al., 2015) COSTA, E. A. et al. CAMAW: A Clustering Algorithm for Multiple Applications in WSAN. In: **International Conference on Systems and Networks Communications**, 10.,2015, Barcelona, v. 1,n. 1, p. 81-89, Nov. 2015.

7.1 Trabalhos futuros

Como trabalho futuro pretende-se incluir no CAMAW capacidades para lidar com os novos desafios das RSSFs. Entre estes é destacada a necessidade de suprir a demanda de um número crescente de usuários por um volume cada vez maior de informações de monitoramento coletadas por infraestruturas de sensoriamento interligadas que podem inclusive pertencer a diferentes RSSFs. Para isso, torna-se importante o desenvolvimento da capacidade de organizar os nós para criar uma instância única da rede a partir de diferentes

infraestruturas de sensores que embora interligadas, são heterogêneas em termos de dispositivos sensores, tipos de comunicação, tipos de dados de sensoriamento, entre outros. Um dos aspectos práticos a serem investigados tem a ver com o aumento do número de nós sorvedouros, fruto das inter-conexões entre diferentes RSSFs e também da maior quantidade de acessos de usuários/aplicações. Estes nós sorvedouros podem ou não representar diferentes grupos de usuários, o que implica em diferentes decisões acerca da escolha da rota que as informações coletadas devem seguir dentro da organização das RSSFs para alcançar de uma forma eficiente o seu destino (usuário/aplicação). A grande questão a ser respondida é se nesse novo cenário mais amplo e complexo o CAMAW permanecerá obtendo as vantagens demonstradas neste trabalho.

Um segundo aspecto é considerar a inclusão de novos parâmetros oriundos das aplicações no processo de organização dos agrupamentos da rede. Entre eles destacam-se a escolha, em tempo de execução, do protocolo de sincronização mais adequado ao conjunto de aplicações ativas. Conforme descrito na seção 5.3, as aplicações podem possuir diferentes requisitos de sincronização dos nós. Como, por exemplo, em Santos et al. (2014) foi constatado que aplicações de SHM precisam de uma sincronização rigorosa para se estabelecer uma alta correlação dos dados obtidos a cada instante por diferentes nós da rede, possibilitando uma avaliação precisa do fenômeno sendo observado. Enquanto que outros tipos de aplicações como, por exemplo, AVAC (aquecimento, ventilação e ar-condicionado), controle de incêndio, entre outros, não necessitam de um sincronismo tão rigoroso entre os nós para se alcançar seus objetivos de monitoramento.

Um terceiro aspecto é a definição de uma nova métrica que torne possível uma avaliação adequada da capacidade do algoritmo em clusterizar as regiões alvo de cada aplicação, em comparação com algoritmos de clusterização tradicionais que, via de regra, clusterizam toda a rede.

Um quarto aspecto trata-se da implementação do modelo de entrega de dados *event-driven*. Sugere-se alterações em estruturas de dados e adição de procedimentos ao algoritmo de forma a habilitar a RSSF no suporte às aplicações enquadradas neste modelo de entrega.

Um quinto aspecto a ser investigado é avaliar a implementação do algoritmo em outras plataformas voltadas para RSSFs. Segundo o trabalho de Maurya e Shukla (2013) e de Vujovic e Maksimovic (2014) as plataformas de sensores MicaZ, IRIS, Raspberry PI e Waspmote são compatíveis com a tecnologia Java assim como a plataforma SunSPOT utilizada neste trabalho. A partir da implementação em outras plataformas de sensores, novos

experimentos podem ser conduzidos para comparar seus resultados com os obtidos no presente trabalho.

Por fim, um sexto aspecto a se investigar é o desempenho do algoritmo variando-se o *node degree* dos nós e o *cluster size* para os CHs. Estes parâmetros limitam respectivamente a quantidade de nós vizinhos ao nó e a força de sinal do CH que delimita a área de cobertura do cluster. Pode ser que em RSSFs densamente distribuídas, não haja a necessidade de envolver todos os nós daquela região, para se atingir o objetivo de monitoramento requerido pelas aplicações. Por isso, estas informações podem ser passadas por parâmetros definidos pela aplicação de modo a facilitar o cumprimento da necessidade de monitoramento das aplicações. Assim, os nós ociosos, que são aqueles que não estão envolvidos no monitoramento, podem permanecer em um estado de baixo consumo para economizar recursos da rede.

8 Referências

ABBASI, A. A.; YOUNIS, M. A survey on clustering algorithms for wireless sensor networks. **Computer Communications**, Amsterdam, v. 30, n. 15, p. 2826-2841, 2007.

AFSAR, M. M.; TAYARANI-N, M. H. Clustering in sensor networks: A literature survey. **Journal of Network and Computer Applications,** Amsterdam, v. 46, n. 1, p. 198-226, 2014.

AIELLO, F. et al. MAPS: A mobile agent platform for WSNs based on Java Sun Spots. In: **International Workshop on Agent Technology for Sensor Networks (ATSN),** 3., 2009, Budapest. Proceedings..., p. 41-48, 2009.

AKYILDIZ, I. F. et al. Wireless sensor networks: A survey. **Computer Networks,** Amsterdam, v. 38,n. 4, p. 393-422, 2002.

AKYILDIZ, I. F.; VURAN, M. C. A Cross-Layer Protocol for Wireless Sensor Networks. In: **Annual Conference on Information Sciences and Systems**, 40., 2006, New Jersey. Procedings… Los Alamitos: IEEE, n. 1, p. 1102-1107, Mar. 2006.

ANASTASI, G. et al. Energy conservation in wireless sensor networks: A survey. **Ad Hoc Networks,** Amsterdam, v. 7, n. 3, p. 537-568, 2009.

ARORA, A. et al. A line in the sand: A wireless sensor network for target detection, classification, and tracking. **Computer Networks,** Amsterdam, v. 46, n. 5, p. 605-634, 2004.

BAO, X. et al. WRECS: An improved cluster heads selection algorithm for WSNs. **Journal of Software,** v. 9, n. 2, p. 507-514, 2014.

BISPO, K. A.; ROSA, N. S.; CUNHA, P. R. F. A semantic solution for saving energy in wireless sensor networks. In: **Symposium on Computers and Communications**, 17., 2012, Cappadocia. Proceedings... Los Alamitos: IEEE, n. 1, p. 492-499, 2012.

BORGES, L.; VELEZ, F.; LEBRES, A. Survey on the Characterization and Classification of Wireless Sensor Networks Applications. **IEEE Communications Surveys & Tutorials,** Los Alamitos, v. XX, n. 1, p. 1-1, 2014.

BOUHAFS, F.; MERABTI, M.; MOKHTAR, H. A semantic clustering routing protocol for wireless sensor networks. In: **IEEE Consumer Communications and Networking Conference**, 3., 2006, Las Vegas. Proceedings… Los Alamitos: IEEE, v. 1, n. 1, p. 351-355, Jan. 2006.

BRANDOLESE, C. et al. Operation of WSNs Periodic Applications. In: **International Conference on Information Communication and Embedded Systems,** 2014, Poonamalle. Proceedings… Los Alamitos: IEEE, Feb. 2014

BRUCKNER, D. Semantic neighborhood sensor network for smart surveillance applications. |In: **International Conference from Scientific Computing to Computational Engineering,** 3., 2008, Athens. Proceedings… Athens: IC-SCCE, Jul. 2008

CALDAS, G. et al. S-Leach: a Leach Extension for Shared Sensor Networks. In: **International Journal of Distributed Sensor Networks,** 14., 2015, Las Vegas. Proceedings… v. 2015, n. 1, p. 661, Jul. 2015.

CHEN, M. X.; HU, C. C.; WENG, W. Y. Dynamic object tracking tree in wireless sensor network. **Eurasip Journal on Wireless Communications and Networking,** Heildelberg, v. 2010, n. 1, p. 8, 2010.

COSTA, E. A. et al. CAMAW : A Clustering Algorithm for Multiple Applications in WSAN. In: **International Conference on Systems and Networks Communications**, 10.,2015, Barcelona. Proceedings… ThinkMind. v. 1,n. 1, p. 81-89, Nov. 2015.

DE C. PASCHOALINO, R.; MADEIRA, E. R. M. A Scalable Link Quality Routing Protocol for Multi-radio Wireless Mesh Networks. In: **International Conference on Computer Communications and Networks**, 16., 2007,Honolulu. Proceedings… Los Alamitos: IEEE, p. 1053-1058, 2007.

DE FARIAS, C. M. et al. A scheduling algorithm for shared sensor and actuator networks. **The International Conference on Information Networking 2013 (ICOIN)**, 2013, Thailand. Proceedings… Los Alamitos: IEEE, p. 648-653, 2013.

DELICATO, F. C. ; Protti, F. ; REZENDE, J. F. ; Pirmez, L. . Uma Abordagem baseada em QoS para Seleção de Nós Ativos em Redes de Sensores sem Fio. In: **Simpósio Brasileiro de Redes de Computadores**, 23., 2005, Fortaleza. Anais… Fortaleza: SBC, 2005. v. 1. p. 367-380.

DELICATO, F. C. Middleware baseado em serviços para redes de sensores sem fio. In: **Congresso da Sociedade Brasileira de Computação**, 19., 2006, Campo Grande. Anais… Porto Alegre: SBC, p. 197, 2006.

DENHOLM, P.; KULCINSKI, G. L.; HOLLOWAY, T. Emissions and energy efficiency assessment of baseload wind energy systems. **Environmental Science and Technology,** v. 39, n. 6, p. 1903-1911, 2005.

EFSTRATIOU, C. et al. A shared sensor network infrastructure. In: **Conference on Embedded Networked Sensor Systems - SenSys '10**, 2010, Zurich**.** Proceedings... New York, USA: ACM Press**:** 367 p. 2010.

FAN, Z. F. Z.; GAO, R. Architectural Design of a Sensory-Node-Controller for Sensor Network. **2005 IEEE Instrumentationand Measurement Technology Conference** Proceedings... Los Alamitos: IEEE, v. 1, n. 5, p. 17-19, 2005.

FARIAS, C. et al. Multisensor Data Fusion in Shared Sensor and Actuator Networks. In: **International Conference on Information Fusion (FUSION),** 17., 2014, Salamanca. Proceedings... Los Alamitos: IEEE, p. 1-8, 2014.

GAO, H. et al. Data collection in multi-Application sharing wireless sensor networks. **IEEE Transactions on Parallel and Distributed Systems,** Los Alamitos, v. 26, n. 6, p. 403-412, 2015.

GAO, R. X.; FAN, Z. Architectural design of a sensory node controller for optimized energy utilization in sensor networks. **IEEE Transactions on Instrumentation and Measurement,** Los Alamitos, v. 55, n. 2, p. 415-428, 2006.

GAVALAS, D. et al. Clustering in Wireless Sensor Networks. **RFID and Sensor Networks**, Cleveland: CRC Press. p. 323-353, 2009.

GNAWALI, O. et al. Collection tree protocol. In: **ACM Conference on Embedded Networked Sensor Systems - SenSys '09**, 7., 2009, Berkeley. Proceedings... New York, USA, p. 1, 2009.

HAGHIGHI, M. An end-to-end middleware solution with multiple concurrent applications support for wireless body area networks. In: **International Workshop on Computational Intelligence and Applications,** 6., 2013 , Hiroshima. Proceedings... Los Alamitos: IEEE, p. 201-206, 2013.

HAGHIGHI, M.; CLIFF, D. Multi-agent support for multiple concurrent applications and dynamic data-gathering in wireless sensor networks. In: **International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing,** 7., 2013, Taichung. Proceedings... Los Alamitos: IEEE, p. 320-325, 2013.

HE, T. et al. Energy-efficient surveillance system using wireless sensor networks. In: **International conference on Mobile systems, applications, and services - MobiSYS '04**, 2004, Boston. Proceedings... New York: ACM, p. 270-283, 2004.

HEINZELMAN, W. R.; CHANDRAKASAN, A.; BALAKRISHNAN, H. Energy-efficient communication protocol for wireless microsensor networks. In: **Hawaii International Conference on System Sciences,** 2000,Honolulu. Proceedings... Los Alamitos: IEEE, p. 1-10, Jan. 2000.

HERMETO, R. T. et al. A distributed algorithm for semantic collectors election in wireless sensors networks. **Journal of Applied Computing Research**, São Leopoldo, v. 3, n. 1, p. 1-10, 2014.

HSIEH, C.-M.; WANG, Z.; HENKEL, J. DANCE: Distributed Application-aware Node Configuration Engine in Shared Reconfigurable Sensor Networks. In: **Design, Automation & Test in Europe Conference & Exhibition (DATE)**, 2013, Grenoble. Proceedings... Los Alamitos: IEEE, p. 839-842, Mar. 2013.

HUANG, P. et al. The evolution of MAC protocols in wireless sensor networks: A survey. **IEEE Communications Surveys and Tutorials,** Los Alamitos, v. 15, n. 1, p. 101-120, 2013.

IZADI, D.; ABAWAJY, J.; GHANAVATI, S. An Alternative Clustering Scheme in WSN. **IEEE Sensors Journal,** Los Alamitos, v. 15, n. 7, p. 4148-4155, 2015.

JAIN, N.; GUPTA, S.; SINHA, P. Clustering Protocols in Wireless Sensor Networks: A Survey. **International Journal of Applied Information Systems,** New York, v. 5, n. 2, p. 41-50, 2013.

JAYASUMANA, A. P.; QI, H.; ILLANGASEKARE, T. H. Virtual sensor networks - A resource efficient approach for concurrent applications. In: **International Conference on Information Technology-New Generations, ITNG 2007**, 4., 2007, Las Vegas**.** Proceedings... p. 111-115, 2007.

KAPOOR, N. K.; MAJUMDAR, S.; NANDY, B. Techniques for Allocation of Sensors in Shared Wireless Sensor Networks. **Journal of Networks,** v. 10, n. 1, p. 15-28, 2015.

KATIYAR, V. A Survey on Clustering Algorithms for Heterogeneous Wireless Sensor Networks. **Int. J. Advanced Networking and Applications,** v. 02, n. 4, p. 745-754, 2011.

KHALID, Z. et al. Middleware framework for network virtualization in SHAAL. In: **IEEE Symposium on Computer Applications and Industrial Electronics (ISCAIE 2014)**, 2014, Penang. Proceedings... Los Alamitos: IEEE, p. 175-179, 2015.

KHAN, I.; BELQASMI, F.; GLITHO, R. Wireless Sensor Network Virtualization: Early Architecture and Research Perspectives. **in IEEE Network,** Los Alamitos, v. 29, n. 3, p. 104-112, 2015.

KHAN, I. et al. Wireless Sensor Network Virtualization: A Survey. **In IEEE Communications Surveys & Tutorials**, Los Alamitos, v. 18, n. 1, p. 553-576. 2015.

LABS, S. Sun ™ SPOT Programmer ' s Manual. **Communication,** v. 0, 2010.
Oracle, Sun Spot Programmer's manual, **Oracle**, Release v6.0, 2010.

LEONTIADIS, I. et al. SenShare: Transforming sensor networks into multi-application sensing infrastructures. **European conference on Wireless Sensor Networks**, 9., 2012, Trento. Proceedings... New York: ACM, p. 65-81, 2012.

LIM, S.-J.; KIM, G.-J.; KIM, D. An Energy Efficient Clustering in Wireless Sensor Networks. **Advanced Science and Technology Letters (ASTL).** v. 95, p. 37-42, 2015.

LIU, P. et al. Semantization Improves the Energy Efficiency of Wireless Sensor Networks. **2010 IEEE Wireless Communication and Networking Conference**, p. 1-6, 2010.

LIU, T. et al. Implementing software on resource-constrained mobile sensors. In: **International Conference on Mobile Systems, Applications, and Services - MobiSYS '04**. Proceedings... New York, USA: ACM Press**:** 256 p. 2004.

LIU, X. A survey on clustering routing protocols in wireless sensor networks. **Sensors (Switzerland),** v. 12, p. 11113-11153, 2012.

MAHAJAN, S.; MALHOTRA, J. Energy Efficient Control Strategies in Heterogeneous Wireless Sensor Networks: A Survey. **International Journal of Computer …,** v. 14, p. 31-37, 2011.

MAINWARING, A. et al. Wireless sensor networks for habitat monitoring. In: **ACM international workshop on Wireless sensor networks and applications**, Proceedings... p. 88-97, 2002.

MARÓTI, M. et al. The flooding time synchronization protocol. In: **International Conference on Embedded Networked Sensor Systems**, Proceedings... p. 39-49, 2004.

MAURYA, M.; SHUKLA, S. R. N. Current Wireless Sensor Nodes (Motes): Performance metrics and Constraints. **International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE),** v. 2, 2013.

MÜLLER, R.; ALONSO, G. Efficient sharing of sensor networks. **2006 IEEE International Conference on Mobile Ad Hoc and Sensor Systems, MASS**, p. 109-118, 2007.

NASIM, M.; QAISAR, S.; LEE, S. An energy efficient cooperative hierarchical MIMO clustering scheme forwireless sensor networks. **Sensors,** v. 12, n. 1, p. 92-114, 2012.

POTTIE, G. J.; KAISER, W. J. Wireless integrated network sensors. **Communications of the ACM,** v. 43, n. 5, p. 51-58, 2000.

RAGHUNATHAN, V. et al. Energy-aware wireless microsensor networks. **IEEE Signal Processing Magazine,** v. 19, n. 2, p. 40-50, 2002.

REYES, J. A. G. et al. Implementation of a Timestamping Service for SunSPOT Sensors. **Procedia Technology,** v. 7, p. 4-10, 2013.

ROCHA, A. R. et al. WSNs clustering based on semantic neighborhood relationships. **Computer Networks,** v. 56, n. 5, p. 1627-1645, 2012.

ROUNDY, S. ; FRECHETTE, L. Energy scavenging and nontraditional power sources for wireless sensor networks. In: STOJMENOVIC, I. **Handbook of Sensor Networks, Algorithms and Architectures**, London: Wiley Interscience, 2005, 552 p..

SANTOS, I., et al. A localized algorithm for structural health monitoring using wireless sensor networks. **Information Fusion**, New York, v. 15, n.1, p. 114-129, Jan. 2014.

SHNAYDER, V. et al. Sensor networks for medical care. In: **International conference on Embedded networked sensor systems, SenSys '05.** Proceedings... New York, USA: ACM Press**:** 314 p. 2005.

SUNDARARAMAN, B.; BUY, U.; KSHEMKALYANI, A. D. Clock synchronization for wireless sensor networks: A survey. **Ad Hoc Networks,** v. 3, n. 3, p. 281-323, 2005.

TAVAKOLI, A.; KANSAL, A.; NATH, S. On-line sensing task optimization for shared sensors. In: **ACM/IEEE International Conference on Information Processing in Sensor Networks**, .9, 2010, Proceedings... p. 47-57, 2010.

TYAGI, S.; KUMAR, N. A systematic review on clustering and routing techniques based upon LEACH protocol for wireless sensor networks. **Journal of Network and Computer Applications,** v. 36, n. 2, p. 623-645, 2013.

VINYALS, M.; RODRIGUEZ-AGUILAR, J. A.; CERQUIDES, J. A survey on sensor networks from a multiagent perspective. **Computer Journal,** v. 54, n. 3, p. 455-470, 2011.

VUJOVIC, V.; MAKSIMOVIC, M. Raspberry Pi as a Wireless Sensor node: Performances and constraints. In: **International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO 2014) ,** 37., 2014, Opatija. Proceedings... Los Alamitos: IEEE, p. 1013-1018, 2014.

WERNER-ALLEN, G., et al. Fidelity and yield in a volcano monitoring sensor network. In: **Symposium on Operating Systems Design and Implementation**, 2006, Seattle. Proceedings… New York: ACM, P381-396, 2006.

XIONG, Q.; LI, X. Cross-layer design of MAC and application semantics in wireless sensor networks. In: **International Conference on Communication Systems and Network Technologies, CSNT 2014**, Proceedings... Los Alamitos: IEEE, p. 147-150, 2014.

XIONG, S. et al. Multiple task scheduling for low-duty-cycled wireless sensor networks. In: **IEEE INFOCOM**, 2011, Shanghai. Proceedings... Los Alamitos: IEEE, p. 1323-1331, 2011.

XU, N. et al. A wireless sensor network For structural monitoring. In: **International Conference on Embedded Networked Sensor Systems - SenSys '04**, 2004, Baltimore. Proceedings... New York: ACM, p. 13, 2004.

XUE, G.; PAN, Q.; LI, M. A new semantic-based query processing architecture. In: **International Conference on Parallel Processing Workshops**, 2007, Xian. Proceedings... Los Alamitos: IEEE, p. 63-63, 2007.

YICK, J.; MUKHERJEE, B.; GHOSAL, D. Wireless sensor network survey. **Computer Networks,** Amsterdam, v. 52, p. 2292-2330, 2008.

YOUNIS, M.; YOUSSEF, M.; ARISHA, K. Energy-aware management for cluster-based sensor networks. **Computer Networks: The International Journal of Computer and Telecommunications Networking,** New York, v. 43, n.5, p. 649-668, 2003.

YOUNIS, O.; KRUNZ, M.; RAMASUBRAMANIAN, S. Node clustering in wireless sensor networks: Recent developments and deployment challenges. **IEEE Network,** Los Alamitos, v. 20, n. 3, p. 20-25, 2006.

ZHANG, G. A. et al. Joint routing and channel assignment algorithms in cognitive wireless mesh networks. **European Transactions on Telecommunications,** London: John Wiley & Sons, v. 25, p. 294-307, 2014.

ANEXO 1 - Nós da rede, suas respectivas coordenadas e unidades de monitoramento

Nó	Coordenadas	Unit 1	Unit 2	Nó	Coordenadas	Unit 1	Unit 2	Nó	Coordenadas	Unit 1	Unit 2
1	(51,92)	5	1	34	(90,24)	3	4	67	(15,5)	5	2
2	(62,99)	1	5	35	(62,60)	2	4	68	(72,87)	3	1
3	(47,42)	2	4	36	(8,50)	2	1	69	(66,41)	4	2
4	(75,75)	1	3	37	(55,28)	5	2	70	(56,46)	1	2
5	(67,52)	2	1	38	(14,40)	3	2	71	(12,24)	4	5
6	(21,34)	2	4	39	(57,2)	1	2	72	(74,13)	4	5
7	(49,49)	5	2	40	(1,51)	1	2	73	(63,83)	2	4
8	(87,99)	4	2	41	(72,66)	5	3	74	(60,11)	4	5
9	(76,79)	2	4	42	(28,79)	4	3	75	(4,85)	4	2
10	(31,26)	3	1	43	(60,13)	5	3	76	(48,39)	4	2
11	(88,64)	5	2	44	(74,38)	5	3	77	(1,1)	1	4
12	(6,13)	3	1	45	(80,84)	4	1	78	(57,23)	4	3
13	(33,71)	1	5	46	(99,80)	5	3	79	(62,1)	3	4
14	(58,60)	3	5	47	(98,98)	1	5	80	(27,37)	4	1
15	(26,74)	5	1	48	(63,16)	5	2	81	(52,7)	4	1
16	(18,90)	4	5	49	(54,16)	3	4	82	(48,91)	1	5
17	(63,85)	5	2	50	(26,4)	2	1	83	(67,89)	4	2
18	(77,89)	3	1	51	(63,76)	5	1	84	(20,12)	2	1
19	(35,90)	3	5	52	(7,36)	4	1	85	(4,92)	2	1
20	(54,97)	3	5	53	(57,18)	1	5	86	(76,49)	2	4
21	(42,26)	3	5	54	(56,6)	3	5	87	(66,10)	5	2
22	(63,77)	3	5	55	(6,51)	5	4	88	(81,81)	2	4
23	(11,4)	5	2	56	(95,92)	3	4	89	(21,28)	4	3
24	(56,3)	2	5	57	(47,23)	3	4	90	(15,78)	3	4
25	(75,36)	3	5	58	(67,35)	3	2	91	(31,74)	2	3
26	(32,15)	4	3	59	(73,10)	4	1	92	(48,66)	4	1
27	(64,95)	2	3	60	(46,6)	1	4	93	(54,48)	2	3
28	(99,78)	5	2	61	(79,29)	1	4	94	(33,14)	1	2
29	(36,93)	4	2	62	(87,67)	5	5	95	(98,79)	2	5
30	(93,19)	5	3	63	(9,93)	2	1	96	(22,46)	1	5
31	(19,33)	1	5	64	(18,80)	3	4	97	(15,15)	4	3
32	(63,27)	3	1	65	(90,71)	1	5	98	(19,56)	5	2
33	(63,95)	5	1	66	(58,45)	1	3	99	(76,12)	2	1
								100	(66,96)	5	2